Resource discovery and Security in Distributed systems

by
Line Larsen

Thesis is partial fulfilment of the degree of
Master in Technology in
Information and Communication Technology

Agder University College
Faculty of Engineering and Science

Grimstad
Norway

May 2007

Abstract

To be able to access our files at any time and any where, we need a system or service which is free, has enough storage space and is secure. A centralized system can handle these challenges today, but does not have transparency, openness and scalability like a peer to peer network has.

A hybrid system with characteristics from both distributed and centralized topologies is the ideal choice. In this paper I have gone through the basic theory of network topology, protocols and security and explained "search engine", "Middleware", "Distributed Hash Table" and the JXTA protocol. I then have briefly examined three existing peer to peer architectures which are "Efficient and Secure Information Sharing in Distributed, collaborative Environments" based on Sandbox and transitive delegation from 1999, pStore: A Secure Peer–to-Peer backup System" based on versioning and file blocks from 2001 and iDIBS from 2006, which is an improved versions of the SourceForge project Distributed Internet Backup System (DIBS) using Luby Transform codes instead of Reed-Solomon codes for error correction when reconstructing data.

I have also looked into the security aspects related to using distributed systems for resource discovery and I have suggested a design of a resource discovery architecture which will use JXTA for backup of personal data using Super-peer nodes in a peer to peer architecture.

Keywords

Resource discovery, DHT, transitive delegation, middleware, heterogeneous networks, network security, hybrid architecture, peer to peer

Preface

This thesis concludes the two-year Master of Science program in Information and Communication Technology (ICT) at Agder University College (AUC), Faculty of Engineering and Science in Grimstad, Norway. The workload of this thesis equals 30 ECTS and the project has been carried out from January to end of May 2007.

I would like to thank, post Doc Ulf C. Carlsen, my supervisor at Agder University College, for excellent supervision and guidance throughout the project period. I appreciate that you came up with an Idea for a thesis when I did not like any of those that were originally proposed. Also I appreciate that you had not planned to be a supervisor at all this year and then you suddenly had two groups to supervise.

Grimstad, May 2007

_____

Line Larsen

Table of contents

Figure list

1 INTRODUCTION

1.1 Background

Computers come in a variety of devices [1] and they can hook up to the internet almost anywhere. This is why distributed systems way of functioning has become so popular the last 10 years. A distributed system like a peer to peer network is perfect for storing and retrieving of data that you or others want to share, but it is not suited today for storing files you do not want others to read. The amount of distinct files that need to be stored in the peer to peer network will be large and only centralized systems are normally built for this kind of task. As a user I would prefer to be able to use a peer to peer network which is open, transparent and free. The individual storage space in the peer to peer network is smaller than the individually fully grown centralized system so if I want to mirror my data onto the free internet, I will have to store the same amount of data from some other persons` mirrored files also. Will I need to buy a larger storing unit for myself or is there a way of using peer to peer distributed network as it is today, possibly only adding a central unit unto it making it a hybrid network?

The definition of a distributed system is that it is a collection of independent computers that appears to its users as a single coherent system [2]. It is possible with a distributed system to use a mobile phone with the feel of being connected to a fully infrastructure based network. To build a hybrid network the hardware is already there, but the communication protocols need to be different and we need to look in to the security aspects.

The USA`s country wide radars were networked together in the 1950`s, which an idea of universal networking. A study from the US Air Force recommended packet switching and in 1969 the ARPANET went live. The rest of the world collaborated on an international packet switched network coming out in 1978. In 1983 the university wide area network became operational and two years later the network was opened to commercial interests. The internet with all its separate networks connected to it paved the way for distributed systems. Peer to peer systems lack dedicated centralized infrastructure, but the internet has shown that it is possible to use internet as the network and just overlay a network of nodes that share the same interest like sharing music files. The free software movement is one such network sharing an interest using the internet as their backbone and using a versioning control system for the versions of different software as GNU with Linux kernel and OpenOffice. Decentralization of information has shown that a decentralized network can manage highly reliable computing due to its openness and autonomous self administered way [74]. Peer to peer systems like Napster, Gnutella, KaZaA and BitTorrent show that excess resources available at the peer hosts can be utilized to support world wide resource sharing. Attaching mobile phones to the internet will allow them in on the file sharing overlay networks also. There seem to be no limit to what we can do; the only limit is the imagination. These peer to peer networks prove also that wide-scale services can be created without relying on any infrastructure, other than the internet itself. From a fault-tolerance viewpoint, peer to peer systems provide a high diversity of nodes with independent failure modes [3].

There has been little interest from companies to adopt the peer to peer design even though a client/server configuration provides a simple architecture and guaranteed performance. They

have not seen the use of it maybe because the peer to peer networks only share resources. It may also be because it seems impossible to earn money on a completely decentralized system. If the industry like music, movie and mobile companies had been a bit foreseen, mobile ad-hoc networks with the services they can provides, might have been an integrated part of peer to peer networks today [4]. Skype peer to peer phoning has been a success and now they are onto peer to peer/internet television.

Peer to peer networks are more scalable, fault tolerant, self-governed and cost effective compared with centralized systems. Efficient query routing is still a challenge and the query routing in peer to peer systems is based on techniques like centralized indexing using a server, flooding, "random walk" or swamping [5]. Structured overlays over the internet peers such as CAN and Chord that are based on Distributed Hash Table (DHT) are effective since there is a structure to the query. They just allow retrieval of a key and value pairs though and are not suitable for full text search. Peer to peer systems are being used to build large-scale information retrieval systems, but current peer to peer systems need metadata such as filename or keywords to perform a successful search. To help with the query, peers with the same interests can form a cluster overlay. This will help so a query can be routed or forwarded to semantically related peers [6]. Another type is a logical overlay network like pSearch which uses dimension reduction (reducing the number of variables) techniques to reduce search cost. pSearch must not be confused with the backup system pStore.

1.2 Thesis definition

The research project will investigate the possibility of designing a hybrid network using the benefits of both distributed and centralized networks.

The final thesis definition is formulated like this:
"*Distributed systems have become widely popular over the last decade. A Distributed system is well suited for storage and retrieval of data/content shared by multiple peers, but is not so well-adapted to scenarios where the number of distinct files stored in the network is much bigger than the number of peers. In such scenarios, a centralised solution risks to become very large and expensive, while a distributed system probably will not be able to guarantee reliable resource lookup.*

There is a security aspect in using a Distributed System for storing your personal files; confidentiality, integrity and availability need to be taken into account.

As part of this project, a survey should be made of different current architectures for resource discovery in large scale systems, such as in popular p2p systems. If time allows the group should also investigate / design a hybrid architecture which achieve reliable resource discovery with minimal centralized server requirements, i.e. which aim to reconcile the favourable properties of both distributed and centralised systems."

1.3 Problem statement

The project will focus on how it is possible to store and retrieve files in computer systems and what topology, resource discovery protocol and security measures we need. A hybrid system based on a distributed infrastructure and ad-hoc networks can be a solution.

Hybrid system solutions exists supporting infrastructure based and ad-hoc networks with middleware supporting the different operating systems (OS) and retrieval / resource discovery protocols, but they are not optimal. Using the two networks together is a challenge.

As a user I want to back-up my files using either my mobile phone or my laptop. I would like to do it any where and not only when I am at home on my wireless infrastructure based network, but also when I am in the airport or even on the train to or from the airport. How will I be able to locate my files when my computer disk crashes? How will the network topology look like and what protocols will the system use to look-up and locate my files instead of some other persons` files? There are companies supporting this today, but there are possibilities of doing this with a peer to peer network also and of course for free.

One issue is how centralized a distributed system has to be to support the amount of files and peers in a back-up network. The question may arise on what is making a system distributed or centralized? Is a distributed system centralized if there is part of the system keeping track of the files like with the peer to peer BitTorrent tracker or does it have to be some central hardware involved, like a server? There is though no doubt that BitTorrent is seen as a true distributed system. What if we have multiple servers spread out through the world in an international company network, will this set-up be a distributed network? The topology or how the nodes or machines are set-up is important, but also how the files are found. What algorithms are used? Will the information flow be too big if one computer or node contacts all nodes at the same time? Is it better to communicate with one neighbouring node which again contacts its next neighbour and sends back information of its findings for either the file or where the nodes are? Should the file be stored as one file or spread like ashes in the wind landing on random surroundings?

To illustrate how we can use such a system, we can simulate that you as a user is travelling or on the move from your office location to dinner in town, meeting up with an old friend. During the meal you would like to show pictures from your last skiing trip to Austria. This can be done today as long as you uploaded the pictures to a central server like a photo company server or to your private homepage area on a leased machine or server. The idea is though that you instead of having a central storing place for pictures and another storing place for documents, you could have access to files from one place only logging in once. It will look like it is from one location, but really the files will possibly be spread to many peers, with you also as a peer offering the same service back to them.

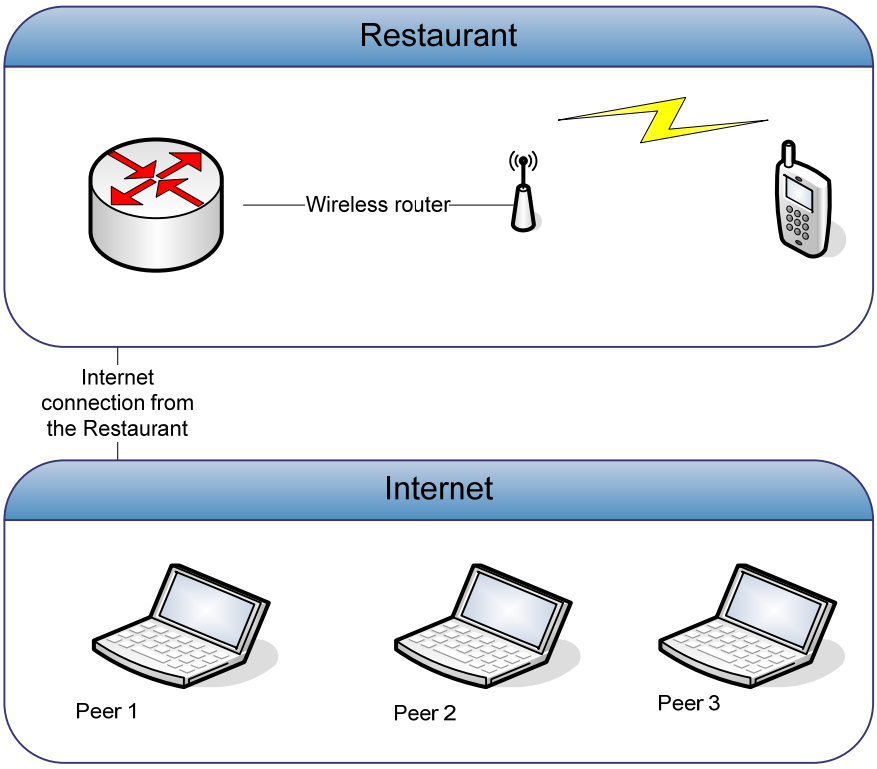

Figure 1 At the restaurant

You have 1000 pictures or files and another user or peer has the same amount. You recognize the problem? You will have to store 1000 files for other peers. There are many files and few users in the system, how is it possible to get around this? A truly distributed system might not be able to handle the scenario. Is the solution to make a hybrid system using the best of both centralized and distributed topologies and with ad-hoc networks attached to it?

What requirements do we have for our future perfect back-up system? Would it be more important that our file is found after our hardware crash than actually how long it takes to find it or is time important? What services do we expect from our system?

- The system will find my file and not some other persons file
- Time is not so important
- No one else should be able to read my file
- I can share a file with someone I trust
- No one else should be able to change my file
- A person I trust can change the file
- If my unit goes down I will be able to retrieve all backed up files
- It must be free
- Enough storage space

Our system should be efficient in the terms of communication, computation and storage utilization and our system should be able to handle queries on demand in a standard operated network. A dynamic topology maintaining an indexing hierarchy will be inefficient and we might need to use content distribution instead of flooding to obtain query results. Flooding is wasting the total bandwidth, though it is a quick algorithm to use for distributing material to every part of the connected network and it is used in the routers today [6].

For designing our system or network we need to know the topology or structure of our network. There are *richly* and *simple* structured objects. This means that with *richly* you can interact with the data and with *simple* you only need to view it like with a web browser on your mobile phone. By supporting the discovery of richly structured objects we would support distributed applications, but it will increase the complexity of the discovery service [7]. What services does the user want? One set of discovery services would be telephone numbers indexed according to names. This might be working for us also searching for a file name. Our system would definitely be heterogenic; no devices connected to the internet are alike and even one user can have different devices he wants to connect to the internet. Then we need a bridging between the different discovery services and middleware possibly using Common Object Request Broker Architecture (CORBA) can do this.

1.4 Importance of study

Most people have their own mobile phone now and it is not only for calling somebody or using sms. The phones are getting smarter and since we are moving from the GSM to UMTS telecommunication system, there is an opening of a wider use of our personal phones and other gadgets that can communicate. Voice is transferred as data packets, so a computer also can act as our phone. This computer to computer communication produces the need for storage space and the need for backing up the data. The data or files will need to be restored so it is important to find which kind of system will be best for doing this. I think the study of Resource discovery ability and Security in Distributed back-up systems is important in our mobile way of living. We are getting used to be able to access information any where we are and at what ever time. It increases our quality of life so nomadic computing is here to stay. Getting to know how this system may work is interesting and important since there is a lot of research and development in this field.

1.5 Report outline

This report is structured as followed:

Chapter 1 gives an introduction to the master thesis which is the current chapter.

Chapter 2 will explain the fundamental theories of retrieval of data and security.

Chapter 3 will explain different peer to peer architectures in use today.

Chapter 4 will explain the security aspects in using distributed system for storing personal files.

Chapter 5 will show an example of a new design of a hybrid system.

Chapter 6 discusses the results in this report

Chapter 7 gives the conclusion of my work and the objectives of this research, also suggest possible further work based on this thesis.

2 THEORY AND STATE OF THE ART

2.1 Theory

File location is a simplified occurrence of Resource discovery [8]. The term resource on the internet was first used to refer to the target of the Uniform Resource Identifier (URL) which is the name of the home page server you visit, like http://www.wikipedia.org. The Resource Description Framework (RDF) which is a Wold Wide Web Consortium (W3C) specification describes URL or the resource in detail. Resources can be static or dynamical. A printer is a static resource and Network bandwidth is dynamic. This thesis is focusing on backing up personal files, so the resources we are looking for will be static addressable documents or files [72]. Network topology, algorithms and protocols and security are important terms for understanding Resource discovery.

2.1.1 Topology

Topology means "geometry of place" and is a large brand in Mathematics. When it comes to computer topology it means how the computers are laid out and connected in a network. The configuration of a computer network can be of many shapes. Two main shapes are centralized and distributed / decentralized.

A person seeing this from an economic perspective said this: "It seems to me that nature operates more along the lines of decentralized redundant/diverse/networked systems but humans often (often for purposes of maximum control and exploitation) build highly centralized systems that are poorly networked, lack redundancy, and are highly centralized"[9].

The more technology or intelligence moved out to the peripheral systems, the more decentralized or distributed the system is. The old telephone network was clearly centralized since there where not much technology in the house phones. Today mobile phones are very smart, but as we know it today, they are still used in a centralized telecommunication system operated by the telephone companies.

Maintenance is easier in an own controlled centralized system. The knowledge is where the main components are and there is higher management control over the system. Not only on hardware, but also on access control lists and other control methods, which are important for network security [10]. A centralized system relies heavily on a few components, which might fail. Few components have a higher risk of creating bottlenecks.

A computer network is recognized by having two or more computers connected together using a telecommunication system for the purpose of communicating and sharing resources. The resource this thesis is focusing on is files. Files can be transferred between any two computers and computers can be desktops, laptops, servers, pda`s or smart phones. There are different ways of setting up a computer network. One is a centralized way.

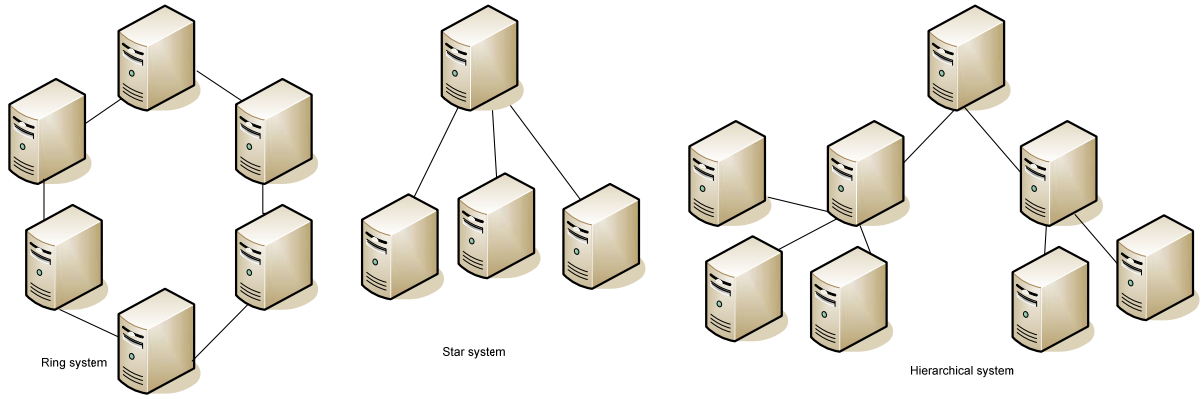

Figure 2 Centralized systems

One centralized system is the ring topology. The ring topology is a way of distributing the server load. The servers are connected and set up looking like one server. The machines are physically close, not placed in different areas of the world and the machines are normally owned by one organization only. The centralized system or star topology is the best known setup. One central server with the client servers connected directly to it. The client can be databases, web servers or other. A hierarchical topology has a central root node on the top of the tree. There need to be 3 levels in this topology since with only one level the system would be a Star system. The Domain Name Service (DNS) on the internet and the Network Time Protocol (NTP) used for timing between computers have both a hierarchical topology.

In a decentralized structure all peers communicate symmetrically and have equal roles, but the peers are not heterogeneous. A centralized system has a hierarchical network node system. A distributed system will have a non-hierarchical network node system or chaos and it will tolerate hardware faults, but can not guarantee communication between nodes [12].

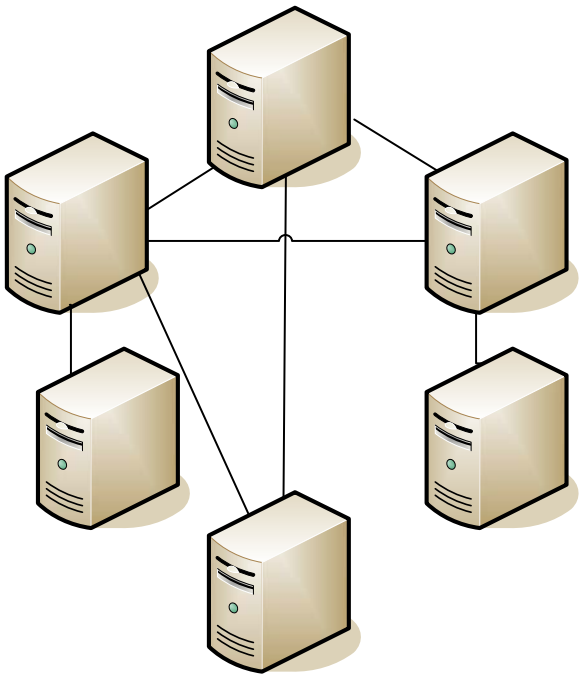

Figure 3 Decentralized/distributed system

In a distributed system no nodes are controlled in any way. The nodes have all equal rights, but the machines are different as for storage space and operating systems. All controlling routines are delegated throughout the network. This set-up is driven by the open source

community; free software and free internet, freedom of speech and freedom from large companies making money on what people now take for granted.

A distributed computer network will not rely on central components. This thesis is focusing on files and file system like Suns` Network file system or Microsofts` Distributed file system will support this. Another name for a distributed file system is a peer to peer (P2P) file system network. ”Edge computing” is another word for the same. “Edge computing” is where the data and computing power is *pushed away from a centralized point* to the edges of the network where you are. The mobile edge of our networks today is getting more mobile, the network perimeter has increased. Something called a mesh topology is also a decentralized network. Mesh topology is used in military application and in peer to peer networks as BitTorrent. In a mesh network all nodes are connected directly to each of the others with wire or wireless.

By using existing distributed system architectures, the systems can tolerate the machines being disconnected like in ad-hoc systems. In an ad-hoc system peers just connect up directly to each other without any infrastructure. This requires automatic configuration and reconfiguration of networked devices and services. A type of network called Mobile ad-hoc networks (MANet) is a self configuring network. Mobile/nomadic computing is previously used with cashing of data, but this is not necessary with the mobile ad-hoc networks we have today [13]. Cashing of data was used for storing data temporarily until the device again was connected to a network.

Another network topology type is the hybrid networks which use a combination of any two or more topologies mentioned above. Hybrid networks are connected in such a way that the resulting network does not look like the hierarchical, star or ring topologies. A tree network connected to a tree network is still a tree network, but two star networks connected together will become a hybrid network topology.

Hybrid topologies or networks may use the best of both centralized and decentralized systems. Real world systems often combine several topologies into one system, making hybrid network topologies. Nodes play multiple roles in such a system. For example, a node might have a centralized interaction with one part of the system, while being part of a hierarchy in another part.

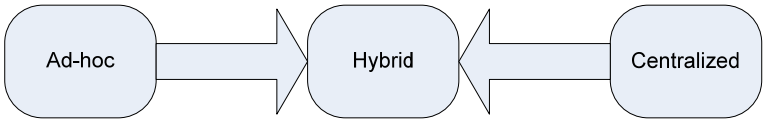

Figure 4 Possible way of making a hybrid network system?

Centralized and *ring* topology can be used together as well in a hybrid system. Serious web server applications can have a ring of servers for load balancing and failover. The server system itself is a ring, but the system as a whole including the clients, is a hybrid. From the clients point of view it looks simple and the ring structure for the server makes it robust.

Then we can use centralized and decentralized systems topology also. Here is where the peer to peer systems come in like with a centralized system embedded in a decentralized system. Peer to peer systems using this set-up are Napster, Gnutella, KaZaA and Morpheus. Internet

email also has this kind of hybrid topology. Mail clients have a centralized relationship with the mail server, but mail servers around the world share email in a decentralized way [14].

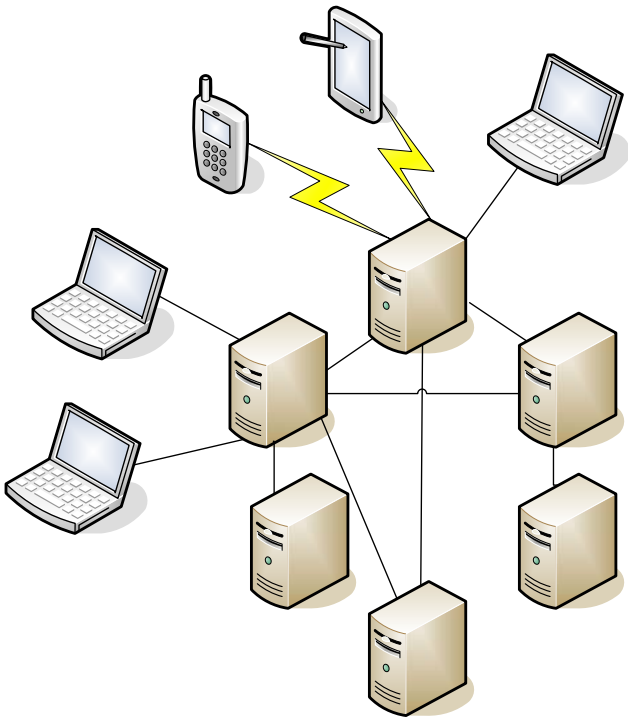

Figure 5 Centralized and distributed system [14]

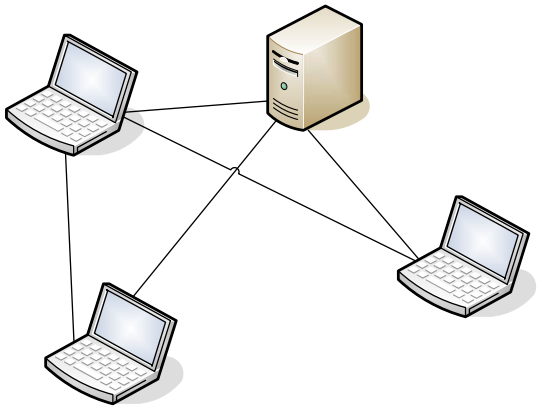

Figure 6 Hybrid architecture [15]

A pure peer to peer network has no servers involved. Pure peer to peer models provide almost plug and play features, since it is possible to use the features of the peer to peer networks just by plugging into internet. Peer to peer will also work for an intranet. In the figure above showing a centralized and a distributed system, the servers are distributed. The Phone, Personal Digital Assistance (PDA) and the laptop connected to one server could be in a coffee shop or at a petrol station where there is a wireless router connecting up to the distributed internet. The laptop could be the one that the person behind the counter is using. Extra to this could be an ad-hoc network where actually the PDA and the Phone communicated over infrared (IR) or Blue tooth creating a Mesh ad-hoc network.

There are also hybrid peer to peer topologies where a server is included. The server would provide a list or an index of already connected peers for the new incoming peer and the resources available with each of them. This will increase the possibility of finding a larger number of peers in the network, but with some problems which will be mentioned. A Super-peer based peer to peer model is suggested by Hao Ding in [15]. The Super-peer based peer to peer model is a pure peer to peer model and a client/server model. A client/server model is a so called two-tier peer to peer model where computers are designated servers or clients. For a standard peer to peer model no peer is designated for anything at all. In a Super-peer based network the clients or workstations run applications and rely on the servers for files, printer devices etc and sometimes processing power, like in a standard network as we know it.

When it comes to distributed programming there are some typical basic architectures or categories. In this thesis I concentrate on the client-server and the peer to peer [74] which are the first and the last one:

- Client-server — Smart client code contacts the server for data, then formats and displays it to the user. Input at the client is committed back to the server when it represents a permanent change.
- 3-tier architecture — Three tier systems move the client intelligence to a middle tier so that stateless clients can be used. This simplifies application deployment. Most web applications are 3-Tier.
- N-tier architecture — N-Tier refers typically to web applications which further forward their requests to other enterprise services. This type of application is the one most responsible for the success of application servers.
- Tightly coupled (clustered) — refers typically to a set of highly integrated machines that run the same process in parallel, subdividing the task in parts that are made individually by each one, and then put back together to make the final result.
- Peer to peer — an architecture where there is no special machine or machines that provide a service or manage the network resources. Instead all responsibilities are uniformly divided among all machines, known as peers.

2.1.2 Algorithm and Protocols

For computers to be able to communicate there need to be a set of rules or protocols. For programs to be able to run, they need to follow a set of algorithms. Both the protocol and the algorithm names are used a bit mixed in this text which can make the reading a bit confusing. A protocol is though the set of rules that the set of distributed algorithms obey [16]. Computing is not what it used to be and the protocols supporting the old thoughts need a review. It actually used to be that the machines in the network were manually set up in a list located on a server by the system administrator. This is still done to some extend, but can not be done with mobile computing where reconfiguration of networked devices and services is necessary all the time. Manual configuration of network IP addresses in this type of network is impossible. There are new protocols for lookup and discovery of networked resources, but still the ideas behind the old protocols are used. The computers need to communicate using one type of protocol, but they also need to follow a set of protocols for lookup and discovery. Different protocols are used for different purposes and some can be used for more than one purpose. I have divided the protocols into chapters of communication, look-up and routing.

2.1.2.1 Communication protocol

When we talk about protocols we normally think of the rules that are set for computer communication, like the Transport Control Protocol/Internet Protocol (TCP/IP) in the transport layer and network layer in the Open System Interconnection (OSI) model, which is the *standard* for computer communication. The Internet Protocol (IP) is below the TCP level in the OSI model. There is a model called “The five layer TCP/IP model” which is a simplification of the OSI layer model. At the bottom of the TCP/IP model we have the physical layer which is the ISDN and modem layer. The data goes from one to the other. Above this layer we have the data link layer which is the wireless standard 802.11, ATM,

Ethernet and GPRS layer. Above this layer is where we have the common set of communication protocols like the Internet Protocol. Above IP we have as mentioned the TCP, but also a protocol called User Datagram Protocol (UDP). TCP guarantees reliable and in-order delivery of data which UDP is not doing, but UDP is faster since it does not check if all the data has arrived, this is also why UDP is used for time sensitive data like with voice and movie streaming. UDP also supports packet broadcast and multicasting. Broadcast means that data can be sent to all which is what normally TV stations do, multicasting is to send to selected subscribers. UDP is stateless, meaning that it does not keep any information on where it sends the data; every data transfer is a new transfer. TCP is stateful, meaning that it keeps track of where in the communication state it is. This is because it also checks that all data has come through and possibly some packets need to be resent.

Above the UDP and TCP transport layer is the application layer where small written applications are placed, like HTTP, SOAP and POP3 etc.

2.1.2.2 Routing algorithms

The algorithms mentioned in [20] are the “Flooding algorithm”, the “Swamping algorithm”, “Random pointer jump” and “Name-Dropper”. The flooding algorithm is used in systems like peer to peer and routing protocols used in routers [21]. The Flooding algorithm can be very slow if it is not started with an initial graph with a small diameter, meaning that is starts with an initial group of node. The first generation peer to peer system like Napster used flooding based mechanism to search desired data. This kind of approach does not work for large systems [22]. The Swamping algorithm is identical to the flooding algorithm except for that it opens up for all nodes connected to it, not only some initial set of nodes. This increases the communication complexity. The routing algorithm “Random pointer jump” would ease on this since it chooses *one* random node to communicate with. Though when using the “Random pointer jump” algorithm the network needs to be strongly connected or else the graph will never converge to a complete graph. A peer to peer network is not “strongly connected” so we will drop the “Name-Dropper” algorithm I think. The Absorption algorithm is another one, but the latest ones are based on the basic and originally ones. A gossip algorithm is another one, however most gossiping algorithms assume knowledge of all the machines that exist on the network or global state which is a lot of information to keep updating all the time.

A [4] push-pull gossiping protocol is another protocol that can be used with a distributed discovery phase first to gather data on peers and to identify popular peers who they call *seers* in a community. This protocol will be used for both finding the peer and finding the resource that the peer has. By using the community of seers it is possible to pull information about the whole community of peers within just two hops. The search can also be used for content instead of file name search.

There are three performance measures for a resource discovery algorithm [19]:
1. time complexity - number of time steps taken;
2. message complexity - number of messages sent; and
3. pointer complexity - number of pointers (machine addresses) passed.

Community search is a way of finding peers and files with a combination of graph theory and link analysis [4]. For instance, if the links on a homepage of a peer were classified as outgoing and incoming links, we would be able to identify a peer to peer community. The links that the peer has stored will show what "interests" this peer has. Since query flooding tends to be very expensive a Semantic Overlay Network (SON) based peer to peer system is an interesting approach. This works like the community search and gossiping protocol. A peer connects to a small set of random peers and queries are propagated along these connections. Semantically related peers have some files in common like "skiing" files. If a peer would like to search for skiing files, it will get a quicker reply from one of these groups. This will reduce the search load.

Another protocol is Broose. It is a peer to peer protocol based on De Bruijn topology. De Bruijn graph can be used to design dynamic networks within the internet so it will work well with peer to peer networks.

Routing protocols have normally been static. If we wish for an ad-hoc topology we can not use the standard protocols that routers use today. Ariadne is a service protocol for Mobile Ad-hoc Network (MANet) integrated in middleware software like WSAMI [13]. A discovery protocol called Universial plug and play (UPnp) sounds like a fun protocol and the goal of this protocol is for all computer devices to connect seamlessly. It is a device control protocol so that all gadgets at home or in the office can be used for communication with standard internet communication protocols. The security protocols are complex for this protocol though. For ad-hoc networks there are two routing protocols; proactive and reactive. The proactive updates the routing table periodically. The reactive produce less network traffic since it only communicates when it is necessary. A combination of these two protocols can be used, then the proactive is used for communicating with nodes close by and the reactive with nodes or terminals further away.

For routing there are different approaches. Gnutella uses flooding which is regarding a bit inefficient in peer to peer systems, but is used in the routers today. More efficient is to use Distributed Hash Tables (DHT), Routing Indicies (RI) and Semantic Overlay Network (SON). The RI is using a way of choosing a subset of peers, rather than selecting a neighbour on random or by flooding the network.

2.1.2.3 Resource discovery (look-up service) algorithms

Within the discovery protocol group, the major difference between resource discovery protocols available is how much they rely on a central directory [21]. Resource discovery protocols for hybrid networks will suit ad-hoc networks. Such heterogeneous environments as peer to peer networks give new challenges to resource discovery. The challenges that need to be looked into are:

- Bandwidth and reliability
- Protocol Interoperability.

To be able to retrieve a file, computers need to follow a certain algorithm. There are different basic algorithms that systems today are based on doing their look-ups. Distributed algorithms

learn of other machines in the network by making queries to machines they already know. The protocols or algorithms also have to be understood by any system or Operating System. We can use middleware to translate the information on the network. Middleware is used for web services and TCP/IP is used for standard communication. Using peers or using a central storage facility, what tool is best for locating our files? Can we use a system which indexes the file *name* or the *content* of the file? How does Google, Yahoo or FAST locate the files we are looking for when we search the internet using their user interface?

Distributed Hash Table (DHT) is the most used algorithm in peer to peer computing. DHT specifies a relation between files and an identifier (ID). Each file is assigned a key generated by a hashing algorithm. The key is then mapped to a node which also has an ID. The DHT algorithm is used in protocols like Content Addressable Network (CAN), Chord, Pastry, Tapestry and Kademlia. "Each DHT scheme generally is pitched as being an entity unto itself, different from all other schemes. In actually, the various available schemes all fit into a multidimensional matrix" [67]. In protocols based on DHT each node is responsible of maintaining the mapping, which makes this algorithm very efficient. The BitTorrent peer to peer protocol is using DHT to look-up each node. The peer to peer protocols have different ways of locating the files, which will be further explained later in this thesis.

Most protocols for distributed hash tables split the key space among nodes according to their identifiers. This result is a very strict topology which is hard to make reliable with regard to node failures. Chord was the first look-up protocol using DHT. The major breakthrough of Kademlia which is another DHT, is to select the nodes storing an association for a given key in a loose manner, or less trict. Other algorithms are Latent Semantic Indexing (LSI) based on Vector Space Model (VSM) [23]. These will be used to classify peers into different categories.

In centralized systems Heuristic (reasoning and past experience) key based routing and standard routing like Open Shortest Path First (OSPF) for routers are used. Z39.50 is another one. It is used for libraries and other information providers. The OSPF use an algorithm called Dijkstra which is based on a "greedy" algorithm. Today companies are using database search technology developed by other companies like FAST. Web crawlers like Google and FAST crawl the web constantly for information and bring the information back to their master. The peer to peer protocols and the Google and FAST protocols operate in different ways.

Libraries used to base their information search on physical cards describing the book and where it was stored. In computing, these cards can be called metadata. There are different metadata schemes. Metadata can describe the content or the resource itself, like the file size etc. Using metadata you will describe the file or data as you want it to be described so that the file or resource can be easily located later, like in a library describing the author etc. Unfortunately different users describe a file differently and the search criteria then need to be different for different libraries or databases. There are definitely other and better ways of finding text today.

A Content addressable storage (CAS) is another way of finding nodes and contents. CAS is a data storage mechanism. It is retrieving data based on its content and not on the location and with this you need at content identifier. Any change to the content will change the content

identifier or address. CAS is best used with contents that do not change often. CAS uses the greedy forwarding algorithm.

2.1.3 Security

2.1.3.1 The CIA of computer security

The CIA of information security is very important because it is the basics of network security.

- Confidentiality (limiting access) measures used: cryptography and digital signatures
- Integrity (data have not been changed) measures used; hash algorithms
- Availability (functioning information systems) measures used; avoid Denial of Service (DoS) attacks [24]

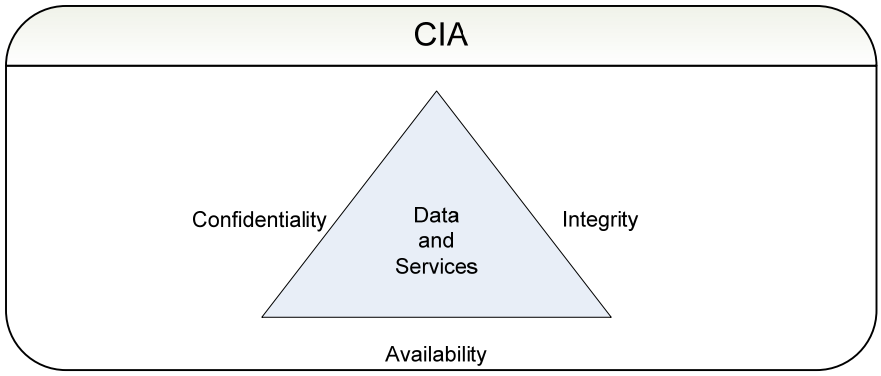

Figure 7 CIA in security

Confidentiality is defined by International Standard Organization (ISO) to be "ensuring that information is accessible only to those authorized to have access". It refers to the need to keep information secure and private. It means ensuring that only authorized parties are able to understand the data. Unauthorized parties may be aware that there is interesting data flowing through the network cables or air, but they should not be able to understand it [26].

In computer security *Integrity* is defined as the prevention of unauthorized writing. No user of a system should be able of making a mistake or be disloyal to modify data items in such a way that information is lost or corrupted, also authorized user. Integrity is defined as the detection and correction of modification, insertion, deletion, or replay of transmitted data including intentional manipulations and random transmission errors.

Availability applies to the flow of data and the accessibility of the system. Information have to be available for those who need it. An attack that makes a system crash will be a problem for the availability of that service. A so called Denial of Service (DoS) attack is when an attacker remotely has access to many computers like a Botnet and use these to access a server at the same time. The server will not be able to deal with all the queries and locks up for some time. In the mean time the competing company server will get the customers that the

attacked server lost due to the Denial of Service attack. Botnets can have 1.5 million peers. As a user we have to make sure that we do not involuntarily become part of one.

In information security confidentiality, integrity and availability (CIA) are the basics. Computer viruses, worms, digital signing, secure communication over wire and wire-less, access control both physical access and software access, cryptography and authentication are only a few words that pop up when thinking of security and computers.

The different types of viruses will not be mentioned here. Physical security of the networks like locks on the doors; is not a scope for this thesis either. The focus will be on the data transmission, but for a service to be available the hardware have to be protected, be maintained and back-ups be done. Data transmission wise Quality of Service (QoS) is important for the availability of the data and services. If you are watching a streamed movie, you do not like it to be interrupted because of the bad transmission lines. Your data or internet service is to be made available on the network, which the Internet Service Provider (ISP) is providing you. The Internet Protocol (IP) packets might be dropped, be delayed or have out of order delivery, data errors might also happen and the availability in CIA is then compromized.

2.1.3.2 Secure data transmission

The TCP/IP version 4 protocol which we still use for communication on the internet, is a totally open data stream. The new version 6 has security built into it and an increased number of available IP addresses for networked devices. For secure transfer of data for version 4 users, they can send the data through secure tunnels like with Virtual Private Network (VPN). Security protocols can be used like IPsec which is integrated in IPv6 and Secure Socket Layer (SSL or TLS). These are securing the network layer or the socket layer. Socket is where the data comes in on the different communication software ports on the computer. IPsec is part of the operating system and SSL is part of the Web application [26] so they cover different layers in the Open System Interconnection (OSI) model. The OSI model is very important for describing the different layers in the computer network communication protocol design. OSI is a Network standard for computer communication, but much taken over by TCP/IP model.

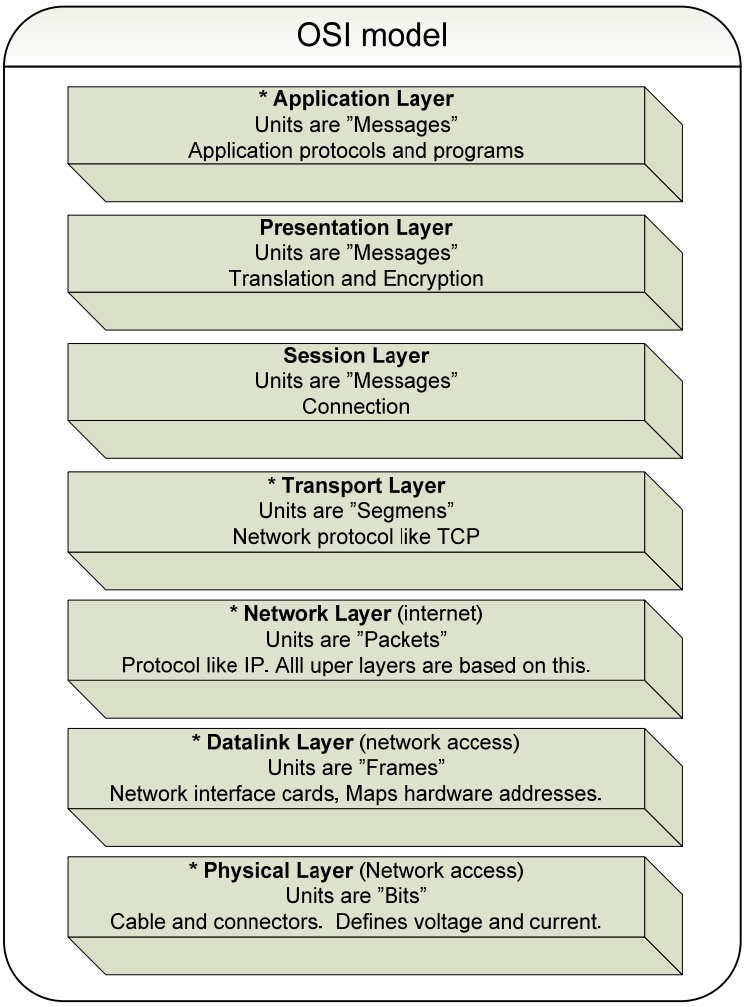

Figure 8 ISO standard 7498

For secure data transfer you can also encrypt the data with symmetric or asymmetric encryption tools or Hash the data. These tools cover integrity and confidentiality in the CIA triangle and cryptography can be used on many levels. For message secrecy we can use symmetric block ciphers. Symmetric block ciphers as; Data Encryption Standard (DES) and Advanced Encryption Standard (AES) which will hide what the data contains. These are working like a digital codebook making the data readable only for those who know the key. Hashing of data using SHA-1 is also a way of securing the data, but also makes it smaller in size. Public key crypto like Rivest, Shamir and Adleman`s RSA do the same. The algorithms from these tools are the basis for security in data transfers between applications. It is though important to have security in all levels. A key used in the lower levels should not be able to leak in to the upper levels [28]. Protocols like SSL and IPSec are though able to secure the key by using certificates approved by a third party certificate authority (CA). Both provide encryption, integrity and authentication [26]. CA issues digital certificates for use by other parties. The certificates contain a public and a private key. There are commercial CA`s and there are government CA`s.

Other ways of securing the information flow is by using message-level security for XML documents [30] called XML signing. Anything that is accessible through an URL can be signed, typically homepage areas on a server. Here the security is embedded in the header or the message using Simple Object Access Protocol (SOAP) which makes it not tied to a

specific transport mechanism or protocol. SOAP is part of something called Middleware which is placed between two applications that do not speak the same language. Middleware acts like a translator for exchange of data and is on the application layer in the OSI model. Middleware will be explained later.

Something difficult to cover on the security side is the "covert channel" which is a hidden message sent parallel on a communication media. Prisoners can send messages hidden in image pixels on a legal internet communication channel or hidden as part of a digital signature of data. The largest problem in internet security is really the "layer 8 problem". There are 7 layers in the OSI model, the top layer is layer 7 which is the application layer. The layer above the application level is the user of the application. It is very important to train the users of the system in network security. Security features embedded in the different layers are of no use if an employee has no moral obligation to the company the user is working for.

An Information Dispersal Algorithm (IDA) can be used for dividing a file into small pieces for security, load balancing and fault tolerance. When you want to send a file, you can not trust it to arrive. By dividing the file in to smaller pieces before sending it, the chance is much higher for the small bits of files to arrive. By adding a bit overhead of pieces of the file, you will guarantee that the file will arrive at its destination. The file will then of course be reconstructed at the destination. The IDA is analogous to the coding scheme used in RAID 5 level backup system. The peer to peer program BitTorrent works like that also. It takes small pieces of a file from a swarm of computers for it to be downloaded quicker. Bits of a file is taken from different peers and downloaded at the same time. This will also prevent congestion in the information pipeline. If you have one large file that needs to be sent, it is more effective to get different pieces of it from peers around the world. Security wise also it is safer to chop the file in pieces so that a man in the middle listening in on the communication channel would not be able to understand the small pieces of it [74].

2.1.3.3 Software security

The operating system (OS) is vital software for the computer. Without OS the computer can not be communicating with the hardware and other software on the computer. OS has access to the hardware, which makes it very vulnerable and important. Only operating systems in government use are fully secure, but OS for standard users today are more secure than they used to be. Many of the insecurities can be fixed by secure programming to avoid "buffer overflow" when the space in memory is too small and "code/command injection" when you can access a command in a normal guest book on peoples/companies home pages. There is a part in the OS called kernel accessing the hardware for the OS application. The kernel is the last frontier of the programmed part accessing the hardware or kernel of the OS like the seed in a nut inside the shell. This is why the kernel programming is the most important. The kernel manages the CPU, memory and devices. The kernel has different access control lists (ACL) for different processes running, so that not all processes would run in supervisor mode with all privileges. A computer does not need a kernel, but it is standard today because it can able the programmers to control the input and outputs. The kernel is also divided up in smaller parts, to be able to address different tasks directly and also secure the different running kernels differently.

Other ways of protecting the computer is using something called Sandbox. Sandbox is a virtual container and used to run untested code or programs from not trusted third parties. For programmers it is an isolated testing environment not mixing with a production environment or operation environment. An application can be tested without testing it live and possibly crash something. Java applets can be a Sandbox and application streaming. You can actually run programs within your web browser or computer, but since they are sandboxed they will just run passively. By using "Sandbox" you will box in the information so that it can not interfere with other programs or have access to your hardware. Reverse sandboxing is when you want to run an application on another computer and you do not want that computer to be able to access your program. It is to extend a private computing environment onto a standalone host in a public environment.

Mobile agents can also be used in security. They are software programs which can be programmed and act on its own on behalf of the user. This is not what we want a malicious user of a network to make. Mobile agents can be used to protect software as for Digital Right Management (DRM). The movie- and music- industry were heavily into DRM, but has loosened the grip this last year since the DRM that have been produced are hacked anyway. A mobile agent can move it self to where the software is, then it can check its status.

2.1.3.4 Secure access

The Access Control Lists are used in programming, but similar can be used to actually access networks also. An ACL is a list of permissions attached to an object or application. It can be what operations that are allowed to access it, or users that are allowed to access it. There are file system ACL`s and networking ACL`s. The Unix file system permissions are like ACL`s permissions; like read, write and execute permissions for each file. ACL`s have been used in distributed systems, but is not adequate [58]. They are used because they are simple, but are originally made for centralized systems where the objects to protect are known. In a distributed system with objects spread over many machines the administration of this is more complex [63].

For controlling the users on a computer system we also have Role Based Access Control (RBAC). In RBAC users are granted membership in a role according to their competence and responsibility in the organization. The operations that the users are permitted to perform are defined according to the role they belong to. Personnel within a department will be given the same role. This is easier than giving each person access rights to different applications that they need to use for their work. Most companies have some kind of security classification as well which works on the file level. A person having a certain role in the company will produce documents that will be labelled unclassified and another one Top Secret [26]. RBAC is newer than Mandatory Access Control (MAC) and Discretionary Access Control (DAC).

Another access control is the context based access control (CBAC). In context based access control, access decisions are based upon the user's context. The context could be time, location of the user, people or technical devices the user is close to, communication channel or strength of user authentication. For example, if the context is the location of the user, access rights are dependent on the network address the user operates from. Compared to RBAC, context based access control is less specific, and it is more like a property than an

access control mechanism. When it comes to distributed networks there is also dRBAC, which is the distributed form of RBAC using public key infrastructure (PKI).

With a "public key infrastructure" a 3.party is a Certificate Authority (CA) guaranteeing that you are being who you say you are. This means the confidentiality part of CIA. When paying for your lottery ticket over the Internet sticking your PKI card into the card reader, a third party guarantees the lottery agency that when you log in and you pay, you are not pretending to be another person. In public key crypto, you will have a private key which nobody knows about and then there is a public key which all know about. It is like if you sign an open letter. The letter is readable, but the signature proves that you wrote the letter. The Smart cards we use in the smart card readers contain a password that we ourselves do not need to remember. This is much safer than remembering passwords. People are terrible at remembering passwords. Using password can grant you access, but it can still not secure the data transfer. Which brings us to the possibility to use the free software Gnu Privacy Guard (GnuPG) which is a hybrid encryption software program using symmetric keys for encryption of the data because it is faster and public key cryptography as well for secure key exchange. There are arranged "key signing parties" where people present their public keys to others. Since they are able to see the person in real life, they will then digitally sign their certificate. The web of trust is an alternative for Public Key Infrastructure (PKI). Higher processing need, reduced bandwidth and disk space is a challenge with PKI. If there are possibilities of eliminating the need for using keys, this will help securing the access of mobile ad-hoc network users.

Kerberos is an authentication protocol built on symmetric key cryptography and is a ticket based system. Windows and Mac OS use Kerberos in a form, but it is dependant on a database storing the secret keys. Unfortunately a system depending on a central unit is not what we are looking for, but Kerberos is wildly used and popular.

Bell and La Padula was originally used to perform access control in military applications based on security levels. A user can only access certain documents that are on the same security level as the user is. The Bell and LaPadula model is a mandatory access control (MAC) opposed to the discretionary access control (DAC). When Bell LaPadula is used with an access matrix, it can be discrete also meaning that accessing a document depends on your role, not only on your security level clearance.

Instead of access rights we can trust objects in another way. Between people we can trust a person we have known for a long time. If that person trusts another person, we would probably trust that person also. The same ways can be used in computer networks. Still I would probably know my friend very well, so I would not trust him/her in everything. The trust would then be related to what tasks that needs to be done. In a computer network you can build up trust. If you have behaved well on the network, you will get trusted. Peer to peer systems will not function if the peers were not able to trust each other. At some point we all need to take a chance. In [48] transitive delegation is one of the tools they want to use for partners that need to share resources. This could typically be some companies working on a joint project. One trusted person or role can delegate roles created by someone else. Transitive delegation is also used in dRBAC mentioned above in addition to using PKI identities to define trust domains [57]. Delegation generally is necessary for scalability of a distributed network. Administrative tasks can then be delegated out to the peers since a centralized unit will not be present in such a network [58].

Transitive delegation or proxy re-encryption is a way of cipher text to be re-encrypted many times. A ciphered text might be re-encrypted from you to a peer and then again from the peer to another peer. The encryption is based on your and the peers key and again then only the peer you sent to and the next peer.

2.1.3.5 Secure storage of data on hardware

Redundant Array of Inexpensive/independent Disks (RAID) is a way of getting information back even though one disk goes down. It is a "data storage scheme". The data is stored on multiple disks and provides data reliability and a quicker read/write process. It is faster to read/write to multiple disks than to only one disk and you will also be able to restore you data if a disk goes down. RAID has different levels, with 0 with no fault tolerance only better read/write capabilities and storage space. The top level is 6 with minimum 4 disks for restoring and spreading the data. The RAID system is used for server systems. The system needs a controller, if this controller goes down it is not possible to rewrite the data. It is then a must to have two of these.

How the RAID levels work with storing data on more than one disk for data reliability and better read/write process, is really how the modern peer to peer networks work also. A distributed and decentralized structure on disks is also good for denial of service (DoS) attacks. A distributed structure will spread the load on more than one machine. So if an attacker is saturating the victim with requests, one disk even though large may not handle all the requests, but many disks acting like one will manage it. Peer to peer servers can though be exploited to be part of an attack on another server, like Botnets which is not safe. For local catastrophes like hurricanes or water flooding, servers need to be protected probably best inside a mountain though communication lines might be flushed away. A distributed approach for servers would here be ideal. Some companies buy network services from companies that can run you system on servers all around the world, it is always nice weather or day some where on the planet. While you sleep in Norway, Australians are awake probably preparing for a BBQ while backing up your data.

When you send data over a radio link or a network cable you want your data to come through. There are different systems to restore data if the link has been bad. NASA use different error coding on their Deep-space telecommunication than what is needed of error coding for the TCP/IP protocol using checksum on the payload. The same techniques used for restoring data on bad data links, can be reconstructing data on hard disks also. The RAID system uses the optimal erasure code which is a parity code that adds a bit, if the data is good or not. This odd or even parity check scheme can only detect odd number of errors. If there are two errors the count of "0" or "1" will make it look like there are no errors in the transfer. A protocol called Reed-Solomon (RS) error correction code is able to reconstruct data when they are encoded as a polynomial known from linear algebra. A newer and better one is the "Tornado" and Luby Transform (LT) code based on a graph called bipartite. The LT code does not need two-way communication which is usually the case for error correction. The RS code is used in Distributed Internet Backup System (DIBS) and the LT code is used in the newer version iDIBS which will be explained later in this thesis.

Hard disk drives crash. One of the reasons is that there are moving parts inside. The disk is rotating like a CD or a LP record. It is not advised to move the hard disk while it is working, since the disk is spinning quite fast. Mechanical moving parts will at some point break down, so backing up data is vital. Flash memory does not have any moving parts and the amount of data you can store on them is increasing. An audio player like iPod has flash memory. Since a moving disk will suffer when you go for a run, a flash drive will not be sensitive for that. A flash drive could theoretically take over for hard disks, but flash drives fail also. Flash memory is though used for running programs directly off it and also flash drives would last longer and store more data than a DVD. So for backups and additional storing capacity, flash drives will help out if we have a hard disk that is about to crash, which can be any time.

Another securing tool is a cryptographic file system where the file system can encrypt a file. The key is only in memory as the file is opened, but this can be enough time for a malicious user to get it if the user is clever enough. These file systems are layered on top of the standard file system and use public-key cryptography.

2.2 Discovery services in use

2.2.1 Search engine

Internet search engines are special crawlers on the Web that are designed to help people find information they are looking for. There are differences in the ways various search engines work, but they all perform three basic tasks [73]:

- They search the Internet or select pieces of the Internet based on important words.
- They keep an index of the words they find, and where they find them.
- They allow users to look for words or combinations of words found in that index.

Early search engines held an index of a few hundred thousand pages and documents and received one or two thousand inquiries each day. Hash tables like DHT reduces the average time it takes to find an entry in an index so today a top search engine will index hundreds of millions of pages. Before a search engine can tell you where a file or document is, it must be found. To find information on the hundreds of millions of Web pages that exist, a search engine use special software robots, called spiders, to build lists of the words found on Web sites. When a spider is building its lists, the process is called Web crawling. The usual starting points are lists of heavily used servers and very popular pages. The spider will begin with a popular site, indexing the words on its pages and follow every link found within the site. Google.com began as an academic search engine. In the paper that describes how the system was built, Sergey Brin and Lawrence Page give an example of how quickly their spiders can work. They built their initial system to use multiple spiders, normally three at one time. Each spider could keep about 300 connections to Web pages open at a time. At its peak performance, using four spiders, their system could crawl over 100 pages per second, generating around 600 kilobytes of data each second. Google, Lycos and AltaVista use different approaches to make the spider operate faster. To exclude the spiders a robot exclusion protocol needs to be added in the Meta-tag section of the webpage. This is used because game pages can mistake a spider for a player [73]. Meta elements are HTML elements used to provide structured metadata about a web page. Metadata was also mentioned

when finding books in libraries. Such elements must be placed as tags in the head section of an HTML document [74]. Sergey Brin and Lawrence Page founders of Google, writes that "it is interesting to note that metadata efforts have largely failed with web search engines, because any text on the page which is not directly represented to the user is abused to manipulate search engines" [62]. Google uses "PageRank" which is a link analysis algorithm that assigns a numerical weighting to each element of a hyperlinked set of documents, patented to Stanford University. A page that is linked to by many pages with high PageRank receives a high rank itself [74].

By using crawlers and *Dynamic Assignment* policy, a central server will assign new URLs (home pages) to different crawlers dynamically. This allows the central server to balance the load of each crawler. The central server may become the bottleneck, so most of the workload must be transferred to the distributed crawling processes for large crawls. There exist two different crawling architectures for dynamic assignment: 1: A small crawler configuration with a central DNS resolver and central queues pr Web site and distributed downloaders. 2: A large crawler configuration in which the DNS resolver and the queues are also distributed. FAST crawler (Risvik and Michelsen, 2002) is the crawler used by the FAST search engine. It uses distributed architecture in which each machine holds a "document scheduler" that maintains a queue of documents to be downloaded by a "document processor" that stores them in a local storage subsystem. Each crawler communicates with the other crawlers via a "distributor" module that exchanges hyperlink information.
Distributed web crawling is a way of parallelization, a large problem can be divided into many small problems, which is what "distributed" is all about.

A mobile agent is kind of like a crawler which is a process that can transport its state from one environment to another [74]. A mobile agent is also able to learn and it can perform actions without requiring continued user involvement. This sounds scary, that it is possible to have something lurking around on your system, just like a crawler does acting on behalf of a user. They are though mostly used for implementing distributed applications in a computer network. Mobile agent applications can also check resource availability and monitoring and be used for resource discovery. A mobile agent can also be used for security purposes, for information flow control. It can probably be used to keep track of peers and what resources they have also.

2.2.2 Middleware

Middleware is very important in exchanging services between different operating systems. Middleware makes it possible to support heterogeneous computers and networks while offering a single system view. Distributed systems are organized in layers of software so the Middleware layer is placed between a higher-level layer of users and applications and a layer of heterogeneous operating systems (OS) [2] [13].

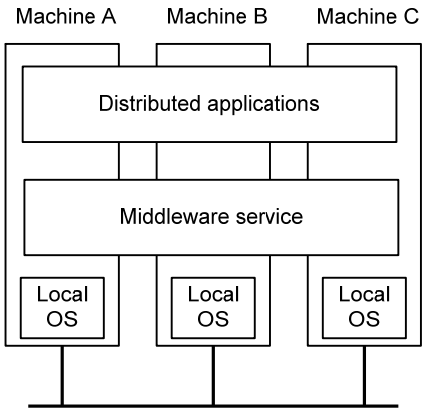

Figure 9 Middleware [2]

Middleware is computer software that connects software components or applications. It is used most often to support complex, distributed applications or programs. It has tools that support application development and delivery. Web services describe a standardized way of integrating Web-based applications using the Extensible Markup Language (XML) , Service Oriented Architecture Protocol (SOAP) , Web Services Description Language (WSDL) and Universal Description, Discovery and Integration (UDDI) open standards over an Internet protocol backbone.

- XML is used to tag the data
- SOAP is used to transfer the data
- WSDL is used for describing the services available
- UDDI is used for listening what services are available

Web services

Format	**XML (format)**	A common format for presenting data. The data can easily be manipulated to meet the presentation of the requestor application.	
Services	**UDDI** A directory service that lists applications that can provide services.	**WSDL** A protocol that enables applications to find a service and to agree on how data and services are to be shared and rendered.	**SOAP** A protocol that enables applications to agree on how data and services are to be communicated.
Network	**The Internet** The Internet, using TCP/IP and other communications/networking protocols, serves as the common network for Web-based applications.		

Figure 10 Web services [26]

Middleware is supporting distributed applications. Microsoft writes: "Web services ease the integration with existing systems. The Web service capabilities offered through like the .NET Framework enable the system to communicate with a variety of back-end systems and desktop applications" [17]. Middleware Infrastructures also makes it easier to manage dynamic networks ad-hoc networks in distributed systems [18].

Available middleware can be classified into four main categories:

- *Transaction-oriented* middleware that mainly aim at system architectures whose components are database applications. Ensure the integrity of the database is maintained.
- *Message-oriented* middleware which is using a message based communication protocol to store, route or transform messages as being delivered.
- *Object-oriented* middleware that are based on the remote procedure call (RPC) paradigm which means a protocol that allows a program on one computer to cause code to be executed on another computer.
- *Service-oriented* system architecture software integration and interoperation (web services) is Small programs (API) that can be accessed over the internet.

There has been development of middleware-based systems though it is said that middleware heterogeneity is still an open issue [35]. Multi-protocol Service Discovery and Access (MSDA) is a research project within the Arles project to overcome the heterogeneity problem in mobile ad-hoc networks. It is layered on top of an already existing middleware, since standard middleware is not addressing the heterogeneity problem. The heterogeneity problem is the different discovery and access -protocols and the connection problems in using different network types [25] like infrastructure based and ad-hoc based. *Service-oriented* system integration provides a mechanism for binding different information systems together at the service layer. With this the organizations can share common system services as well as information. System services may be shared either by hosting them on a central server or by accessing them through distributed objects or standard Web services mechanisms [26].

2.2.3 Distributed Hash Table (DHT) algorithm

DHT is a look-up service. It is part of the third generation of peer to peer overlay networks. The first one Napster, depended on a central database. The second one was Gnutella based on the flooding algorithm. The new one BitTorrent is based on DHT. A Hash function compresses the information. Hashing is also used in cryptography for authentication and message integrity. Hashing is also used in checking if all data has come through correctly for a Cyclic Redundancy Check (CRC). Since the data is compressed into a Hash key, it is possible to compare the two keys. The sender sends the hashed key with the open data and the receiver hashes the data received and compares it with the sent hash. If they are alike, there is no data missing. A Hash key is the unique identifier of the data item. In Distributed Hash Table (DHT) based peer to peer systems, files are associated to keys [22] not data packets in production networks. The DHT functionality allows nodes to put and get files based on their key which is very useful for large distributed systems. In DHTs, each node handles a portion of the job load which is positive for the load balance. In a centralized system one server is doing the whole job, while using DHT the job is divided among many

peers [34]. DHT is so good because no node needs to know about the state of all other nodes, called "global knowledge", since the load is divided among all the peers there is no "single point of failure" either. With many nodes doing the work, DHT scores high on scalability.

DHT was introduced in 2001. It is a table that maps keys to data values. Knowledge of DHT algorithms is a key ingredient in developments of distributed applications [36] because DHTs can scale to extremely large numbers of nodes and to handle continual node arrivals, departures, and failures. Chord, Kademlia and Pastry are DHT`s and look-up protocols like BitTorrent is using DHT. Unlike existing master/slave database replication architectures, all nodes are peers that can join and leave the network freely. In a DHT, the insertion and removal of nodes is independent of the insertion and removal of data, since the data is still stored in the network of nodes even if one node logs off for the day.

- There is no center server in a DHT Network.
- Every client takes charge of a small range of routing
- Every client has a small set of data storage
- In the whole DHT network, data can be found, read and written even without all the nodes connected.

DHT is used by peer to peer systems which are categorized as structured. CAN which is the first distributed hash table before Chord and Kademlia, is "structured". This means that there are some laws on were the peers are. In an unstructured network, information needs to be flooded since there is no knowledge of where a peer with that certain information that we need is. Flooding will increase the information flow in the network. Since DHT is structured and does not use flooding, DHT is more effective. The CAN protocol uses the greedy algorithm instead. Using a structured approach, DHT systems can be used to perform efficient filename searches because they guarantee the location of the data [4]. DHT assigns each file name with a key generated by the hashing algorithm, then it maps the key to the node or peer which also has an ID which can be a hashed IP address [34].

2.2.4 JXTA protocol

Juxtapose (JXTA) is a hybrid peer to peer protocol specification that includes network addressing, routing and messages [39]. JXTA technology is based on a set of open peer to peer protocols. It promises that any device with a digital heartbeat can connect to it. They also have addressed security from the beginning. It was created by Sun Microsystems, but is now an open standard. It will allow communication even if a peer is behind a firewall and it is platform independent meaning that it can run on different operating systems (OS). With JXTA you are supposed to still be able to be contacted even though you are logged off the system [38]. Also a user or node gets a unique address which does not change with the IP address when he or she is on the move. Juxtapose means "side by side" which is a rewriting of the term "peer to peer". JXTA uses Super-peers which can operate as a proxy server for peers that are on the edge of the network overlay, with low bandwidth and bad connectivity. The Super-peers are divided up in rendezvous and relay peers. If the network is divided on different subnets, there will be at least one rendezvous peer. The rendezvous peer is in charge of coordinating the peers in the network. A relay peer allows a peer behind a firewall to be part of the JXTA network. JXTA uses Java based protocols. Java is also a product from Sun Microsystems. JXTA also use DHT and not crawlers for searches. JXTA search is suitable

for environments where content is rapidly changing. JXTA is licensed under the Apache license and JXTA is used in companies like Nokia and Jet Propulsion Laboratory (JPL) which is NASA's lead center for robotic exploration of our solar system. Nokia's Automatic Network Services (ANS) uses JXTA to communicate between the individually installed machines. The ANS group specializes in secure content management. The JPL's software will be used to support data distribution for future planetary missions. The same software is being used to simulate how cancer works so that it can be discovered in an early stage. Security is important in JXTA. In the bottom layer of JXTA there is the JXTA Core where the functions for security are located. On top of this layer there are JXTA services which are indexing, search and file sharing. There is also an application layer which makes it possible for programmers to build peer to peer networks on top of the JXTA protocol. Other peers will get access to the network via "advertisements" which is an XLS document. For connecting PDA's etc to the peer to peer network there will not be need for a driver. A PDA or a mobile phone only need to be compatible with JXTA [71].

2.3 Requirements for a centralized system

In a centralized control network, there is a single server carrying out synchronization of the different clients and the causality control. In a peer to peer centralized system this is applied using an index server to store all the data address information. When a node makes a request for a data item, it queries the server and it returns the desired data address, then the node accesses the node storing the data, directly.
This typical system is Napster. It works by operating a central server which directs traffic between individual registered users. Each time the user submits a request for a song; the central server creates a list of users who are currently connected to Napster whose collections include the specified song. Although centralized peer to peer systems are simple and easy to manage, they have some obvious problems which are not suitable for large scale systems:

- Single point of failure
- Giant communication traffic and storage on server

In a fully centralized system a centralized messaging system will consist of a large data centre that hosts many servers like global catalogue servers, domain controllers and exchange servers (electronic mail, shared calendar, data storage and support for the mobile and web based access to information). The data centre will support all messaging systems for the users, whether they connect locally or remotely.

The characteristics of a centralized messaging system is [41]:

- Data is hosted and managed in a centralized location regardless of whether the users are connected remotely.
- Software upgrades can be rolled out from a centralized location.
- The data centre incorporates power-insulating devices such as an uninterruptible power supply (UPS) so that it is possible for a controlled shut down of the servers and "hot site" (get the system quickly up and running after a disaster) or "cold site".
- Business requirements associated with reducing cost and security requirements are usually the driving forces behind centralizing systems.

If you are planning for a centralized design you should consider it if:

- If you have access to fast network connections between the data centre and your clients else where
- If you save money on administrative and operational costs by centralizing compared to the how expensive it is to install high-end servers and computer clusters in the data centre
- If single point of failure is no problem
- If you have the possibility of caching data for remote users if there is a network outage
- If it is ok to have a lot of reliable storage systems with the large centralized data volumes.
- A centralized model gives you easier security management and greater degree of control over the system physically and data wise.

Google is using centralized indexing. In "Anatomy of a Search Engine" Brin and Page have the opinion that there will be no problem indexing everything everyone in the US has written for a whole year using a Centralized Indexing Architecture, because storage space is now cheap [62]. Will this be the future? Do we not need to use the peers in a peer to peer network for storing our index or even whole files?

2.4 Requirements for a distributed system

Early decentralized peer to peer systems are not based on a central server. They use "broadcast" to search the desired data. When a node receives a request for a key which represents the data item, it attempts to retrieve the file locally if possible. If this is not possible it forwards the request to another node. When a request is successful or failed, the desired data item or failure report is returned to the requester along the same path of the incoming request. Some systems use caching or forward the request to neighbours that are more likely to store the desired data. The typical systems are Gnutella and Freenet. They use the flooding algorithm and are classified as "unstructured". The disadvantages of this system are:

- No guaranteed reliable content locating information
- Expensive cost for routing (flooding mechanism) and scalability

There are some challenges in peer to peer systems. Two main lessons from a measurement result [42].

1) Any similar peer to peer system must be very careful about *delegating responsibilities* across peers (since they are all alike)
2) Peers tend to deliberately misreport information if there is something to gain from it

Because effective delegation of responsibility depends on accurate information, it means that future systems must have built-in motivation or reward for peers to tell the truth. Systems must be able to directly measure or verify reported information because peers will not report that they have a high speed link and have a lot of CPU power. A peer that reports high bandwidth is more likely to receive download requests from other peers, consuming network resources, which is not positive for the peer. The peers are different, so we need to get

accurate information on these two issues. In peer to peer systems, files are stored on the computers of the individual users or peers and exchanged through a direct connection between the downloading and uploading peers, over a standard web browsing HTTP protocol.
All peers in this system are symmetric ie they all have the ability to function both as a client and a server, but they are not alike. Their heterogeneity is a problem when they do not accurately report bandwidth, but by using meta data which is already available it is possible to get bandwidth, number of files currently being shared by the peer, current number of uploads and downloads in progress, names and files being shared and the IP address of the peer. This is possible with the BitTorrent client application Azuresus. Problems with peer to peer networks are 1) free riders who only download files and never keep their client program open so that they upload also 2) no proper connection between peers since they are more or less ad-hoc 3) the heterogeneity is a problem. The last can be fixed by delegating different degrees of responsibility to different host depending on the peers' characteristics like bandwidth, cpu power and storage capacity and also the degree of trust the peer has in the network, but we need to know this information before we delegate responsibility.

2.5 Centralized versus Distributed

The peer to peer topology is very appealing, since it is so accessible and familiar. A centralized system is more like something a large company would use and mixing this with a peer to peer idea that everything can be distributed is disturbing the glorious view. Indexing can be distributed, timing of the computers can be distributed and there is storage capacity out there on people's machines. Trust is functioning in peer to peer systems today and security has become a natural part of applications running on machines in everyone's homes.

Early [44] they could see that both centralized and distributed computing could be used together in a network. The problem they had was that there were no protocols made for communication, file structure and databases for this. Today we have the technology, but "control" is still an aspect when it comes to which system to use.

Students still in and just come out of universities have produced peer to peer systems, like Gnutella and BitTorrent. These systems are also used by serious companies today. The music- and movie- industry has finally figured out that the peer to peer systems are a good tool for distributing their products.

The distributed approach has advantages:

- Reduced core network bandwidth requirements
- Improved response time
- Decreased jitter (unwanted variations of signal characteristics)
- Increased reliability [44]
- Small disks are effective when data is needed often

The centralized approach has advantages:

- Better network communication
- Better synchronization between nodes
- Better latency

2.6 "Resource discovery and Security" literature review

Resource discovery and security has been an issue from the start of computer networking. It is interesting to read about United States Air Force computer security technology study from 1972. A new and also interesting is the "Peer to peer communities: Architecture, information and trust management" by Mujtaba Khambatti which is a dissertation for the degree of Doctor in Philosophy at Arizona state university. The span seems long from USAF in 1972 and Philosophy study in 2003, but the ideas are the same. Literature which is more descriptive is the two on the DHT algorithms Chord and Kademlia from Massachusetts Institute of Technology (MIT) and New York University (NYU). Most of the references are from the USA, but there are others also. Most of the Literature reviewed was found in the databases IEEE, ISI and ACM. We are encouraged to use the reports in these databases, so when I have searched for a topic it does not matter which country the papers come from. Two references I have used are from Norges Teknisk-Naturvitenskaplige Universitet (NTNU) which are a master thesis in "A Security Focused Integration Architecture for an Electronic Observation Chart" in hospitals and a dissertation for a Doctor's degree on " A Semantic Search Framework in Peer-to-Peer Based Digital Libraries. Another one is from Wuhan University in China where our fellow students come from which is on "Efficient Query Routing for Information Retrieval in Semantic Overlays". I have also used French, Indian, Italian and Japanese references in this thesis. When seeing that people all around the world are interested in the same issues, I see the point of internet and the importance of sharing information. Though the most important reference I have used, which explains to me when there are something in the other papers that I do not understand, is Wikipedia. Wikipedia is updated by individual who also write the papers that I have read, but they then pay more attention to explaining details on a subject. Together with the official databases, Google has also been of very good help. I am really impressed of how much information their databases are holding.

3 SURVEY ON DIFFERENT CURRENT P2P ARCHITECTURES

3.1 Peer to Peer systems

One common characteristic of peers in a peer to peer network is that they exist on the edge of the regular network. Because they unpredictable connect and disconnect themselves and with then variable network addresses every time, they are outside the standard scope of the Domain Name System (DNS) which is the internet naming system translating IP address to names of machines that humans can read and remember [31]. How can we deal with such nodes, which at one point are connected and then suddenly gone?

Peer to peer protocol overlay networks can be categorized after their generation or by being centralized like the first peer to peer protocol Napster, decentralized like KaZaA, structured like CAN using DHT, unstructured like Gnutella using flooding or hybrid like JXTA.

Napster, Gnutella, KaZaA and BitTorrent are peer to peer download sharing technologies [32]. The newest is BitTorrent. Napster uses a centralized registry with a well known IP address that users query for song selections [33]. This centralized topology was what made Napster easy to target legally. It is though now running legally, but not sharing music for free. Gnutella uses a proper peer to peer architecture. The requests get forwarded to the peers and the peers forward the request further until the resource is found or the time to live (TTL) value is reached. The peer to peer system Gnutella is one of the most pure decentralized or distributed systems. It has only a small centralized function to bootstrap a new host ie starting up a new host so it can work within the network. KaZaA is another product. Those that developed KaZaA also developed Skype IP telephone and Joost IP television. BitTorrent is though the discovery look-up service which is able to download files faster than any other software device. This is due to a file is downloaded from many peers at the same time. One part of a file is downloaded from one peer and another part downloaded at the same time from another peer. *Swarming* (group of peers connected to each other) technology is used in BitTorrent, instead of treelike network structure [73] in standard file systems. BitTorrent has become house-broken and Paramount, MTV, 20th Century Fox and Warner bros have their movies on the BitTorrent homepage. There are some points to BitTorrent that makes it better than the rest of the peer to peer protocols:

- BitTorrent makes many small peer to peer requests over different TCP sockets, while web-browsers make a single HTTP GET request over a single TCP socket.
- BitTorrent downloads in a random or "rarest-first" (rarest piece of file [64]) approach that ensures high availability, while HTTP downloads in a contiguous manner.
- BitTorrent is using both trackers (centralized servers) and DHT (Kademlia)

Another decentralized network is Freenet. It has a heuristics key based routing, remembering using reason and past experience. Freenet is a censorship resistant peer to peer network. No nodes are rated in Freenet which is unusual for a peer to peer network, but probably so because of the peers supposed to be anonymous. Freenet does also key based routing like DHT, but it is not the same. Freenet does not guarantee that it can find a given piece of data.

A routing algorithm is Kademlia. Kademlia is a routing algorithm using XOR metric routing, based on Distritbuted Hash Table (DHT). XOR function returns zero if bits are the same and one if bits are different. Kademlia searches the network bit by bit to finally become close to zero which is the best node. Kademlia uses the User Datagram Protocol (UDP) which is on the same level in the OSI model as the Transmission Control Protocl (TCP) layered above the Internet Protocol (IP). The Kademlia algorithm is based on calculating the distance between two nodes. The distance here is the exclusive OR (XOR) of the two node ID`s. Kademlia is used in BitTorrent. Other DHT`s are Chord, CAN, Pastry and Tapestry. Chord was the first DHT algorithm and bases the search on length r = O(LogN) as Kademlia where N is number of nodes in the network and O is just describing the function. Chords length is not the distance between two nodes, but the number of nodes traversed during a look-up operation.

There are different approaches to how to design a peer to peer network as those mentioned above. There are different protocols in use to find each individual peer. In a client/server-based model, any peer looking for resources does not need to visit other peers than the server itself since the server is maintaining the information of each peer. The server can be a discovery, lookup and content server or taking care of only one of them or two or all three. The server will though slow down if there are too many requests generated simultaneously.

Single point of failure does not exist in peer to peer models. In a distributed environment a peer to peer network that lacks a central authority, a simple formation and discovery problem might become complex. In order to compensate for the server, the cost of communication will be higher. A network called a "Super-peer" peer to peer network is an integration between the pure peer to peer model and a client/server model. A node is given a Super-peer tag and will operate as both a client and a server to a set of clients and as an equal with a set of other Super-peers. The Super-peer or group of Super-peers will provide services like listing of connected peers, acting as primary connection node and sometimes operate as a search hub. Together the Super-peers will form a pure peer to peer overlay network. A practical system in architectures like this is JXTA. It provides the efficiency of a centralized network and also autonomy, reliability and load balancing of a distributed network. An overlay topology like de Bruijn also uses the Super-peer structure [22].

3.2 Efficient and Secure Information Sharing in Distributed, collaborative Environment

The idea behind this system is when a set of partners need to share resources [48]. The partners could be different departments or organizations. I vision a company like Aker Kvaerner and its people who are doing work for Statoil on a project. In the project people from both companies will have to be able to access each others files etc, but only the files or areas related to the specific project. The paper suggests the use of transitive delegation, cryptographic file system, capacity sandboxing, reverse sandboxing and fine-grained access control. The collaboration model they use is similar to Common Object Request Broker Architecture (CORBA). The access control is controlled by agent (an engineer), asset (host) and object (program) certificates which include the holders` identity, key and information based on if you are an agent, asset or object. A delegate is an activity created by an agent and it can access multiple objects. An agent can create a delegate and a delegate can create another delegate and these can be created on the source or on a distrusted target machine or asset. Transitive delegation is an explanation of a goal to allow a delegate running on a not trusted asset, to be able to create a delegate on another asset. It is like if I trust a person, I also trust that persons friends. The cryptographic file system (CryptoFS) they use has a cryptographic key for each file. They want to use this system so that the servers or storing units for these files do not need to be secured in any way. There is no authentication or access control on the units the files are stored on. The key for each file is stored in a "key server" which can be an asset on some partners' hard disk or "server". To get to the file though, the participant or agent will need to authenticate it self to the "key server" using its certificate. The files can not be updated, only changed out and then with a new key associated with it. With this set-up both authentication and access are separated. This can also be done to other tasks than to get a file. If you want to run a program or maybe even a movie, you get the "key" for the movie you want to watch at the key server and then you have the key to be able to run and stream the movie to your machine and screen. The certificates are changed out often, but the private keys stay the same. A Certificate Authority (CA) is dealing with the keys. For secure deployment of delegates to assets, only partly trusted sandboxing and reversed sandboxing are used.

3.3 pStore: A Secure Peer To Peer Backup system

This system is based on a peer to peer system overlay and they specify that they are doing incremental backup also [54]. Here they though mean backing up the files that are updated, leaving the files that are not changed. Software developers use a system called Concurrent Versioning System (CVS) which helps developers specially those that work on open source software were anybody can update the software, to know which version of the software they work on. Versioning can also be used in backup systems. The versioning type pStore is using is rsync type for Unix OS systems. They use Chord to locate the peers to where the files will be stored. The files are not stored as a whole, the files are split into chunks or they use the "exact-copy chunk replication" and the chunks are spread to different peers. The chunk is called a File Block (FB) and the File Block List (FBL) keeps track of the ID, hash, length and offset. The FB and the FBL are encrypted with symmetric keys to preserve privacy. The FBL's key is derived from the users private key. Each chunk of data includes the owners` public key, so anyone can view the metadata of the file chunk or FB. Only a small number of replications are needed to ensure an acceptable level of redundancy. A quota policy where the amount of space available for a user is proportional to the space contributed to the system is a wished part of pStore, but they did not implement a mechanism for it. RSA is used for public key encryption, EAS is used for symmetric key encryption and SHA-1 for cryptographic hashing. With pStore they discovered that with digital signatures they found that performance, bandwidth and storing got worse. Not surprisingly.

3.4 iDIBS: An Improved Distributed Backup System

The original Distributed Internet Backup System (DIBS) used Reed-Solomon error correction code, but iDIBS is using a new type called Luby Transform which is supposed to be better for large files [56]. LT coding only needs one-way communication to get the data across to the receiver end. These codes are used for restoring the data since in peer to peer networks the peers can join and leave the network at any time. DIBS is a sourceForge project [55]. One of the objectives of both DIBS and iDIBS is that it is important to make backup cheaper. This is probably why DIBS and iDIBS are based on peer to peer systems and the internet. There is no concern about the amount of disk space available in a peer to peer network, which is also what the Microsoft group agrees with [68] and also the founders of Google [62]. iDIBS has reliability and recovery as their major concerns. A measure to identify the peers which are storing your data, after your disk has crashed has been solved by storing some information about the other peers on each peer. Another is that some files need to be updated quite often and this will create network traffic and the error correction can increase the processor time needed. The packet transfer mechanism in iDIBS is improved. A backup of a peer list is incorporated in iDIBS and only one list is needed to recover the entire peer list. The list is backed up every time a peer changes it. The peer list is stored on each peer. The data is encrypted using Gnu Privacy Guard (GPG) which is an open standard and GPG is also used for signing. The users' GPG key will though have to be stored offline on a CD or USB card.

4 SECURITY REQUIREMENTS OF A SYSTEM

I have divided this chapter in three. First part will be on "Security policy", the second on "Anonymity and Trustworthiness" and the third is on Physical security.

4.1 Security policy

Network security is not something we can just apply on top of a system. Network security will have to be built into the foundation of a computer system and also into the layers on top of this. The network infrastructure, the policies and the network access points combined together form the security chain where each link in the chain is important.

For ad-hoc distributed systems, security is a challenge. This is because users are connecting up their mobile units anywhere at any time. An ad-hoc network does not have a central infrastructure to rely on for securing the network. Instead a decentralized trust management can be used. Also cryptographic protocols in use on desktop computers can not be used on mobile devices that most of us use today, since they need more processing power and network bandwidth than mobile devices have access to today[18].

Ideally we do not want a central component in our network. A fully distributed system is what we dream of.

Our objectives are:

- Secure data transmission (Protecting network traffic)
- Software security (delegating rights)
- Secure access (Authenticating the communicating parties)
- Secure storage of data on hardware (Enforcing access control policies on the objects' member functions)

Peer to peer networks still have a lot of security issues.

We can expect attacks like:

- Attack on plaintext/ciphertext
- Attack on protocols
- Attack on access control mechanisms
- Denial of service (DoS) attacks

Trust and policy management are really important for computer security. If there are no policies, it is impossible to be able to set up a system according to "nothing". Most people have an idea of what should be protected like the points above, but they need to be put into policies. These policies can then be followed up by programmers, network administrators and users. Some points could be:

- Criteria for trust
- Confidentiality policies need to be modelled
- System composition
- Covert channel analysis (hidden channel)

4.2 Anonymity and Trustworthiness

Anonymity or trustworthiness in Distributed systems [75] could be possible. There are networks that provide you with anonymity like the Tor onion routing network. For trustworthiness it is usually tied to digital signatures or other form of authentication. Though a signature has only value for how much we trust the verifier of the signature.

There is no central, trusted authority in distributed systems. In an operating system there is a unit controlling the interaction between users and processes. A distributed system is a network overlaying the inter-network, where the peers or hosts will run different operating systems with different security policies [45]. The security functions distributed out in the system need also to trust the other parts of the system to be able to perform their own security functions [46]. We possibly need a *reference monitor* controlling the interactions that an OS has between users, applications and the application's data. The function of a reference monitor is to control and authorize accesses made by subject to objects.

- The users and applications are subjects; they are active since they can manipulate data.
- The application's data is an object; it is passive and is holding the data.

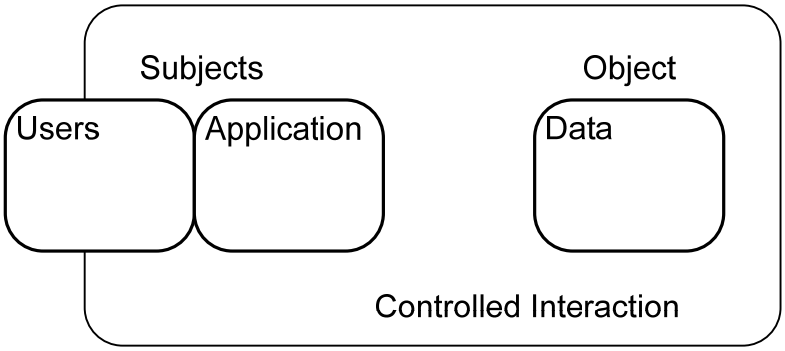

Figure 11 Reference Monitor

A reference monitor is a module that controls all software access to data objects or files. The newer operating systems (OS) have this, like for Microsoft products it is Windows NT/2000/XP and up. This issue has been important for a very long time. In 1972 the United States Airforce (USAF) made a study on planning of computer security technology [47] and point out in the report that IBM at that time planned to spend 40 million dollars in a 5 year period on the computer security problem. The study makes clear that it is the "malicious user threat" which is the single largest barrier for providing multilevel secure processing. One of the issues this study point out is that they want; "controlled execution of a users program or any program executed on a users' behalf. We explicitly include the operating system service functions in this requirement". They also ask for a "reference validation" mechanism which today can be called a reference monitor. The Bell and LaPadula model comes from this research [59].

Trust is very important and each level will need to trust the level below. Even on the lowest level there need to be security measures. This will make all systems and applications secure

based on the basics. Something called the Trusting Computing Base (TCB) is hardware, firmware and software components that are classified as critical to a computer system. The TCB is one of the main targets for attack countermeasures. We can not expect the network or system to be secure if the base is not secure. Each link in the security chain is important. The TCB depends though on the security policy issued, since it is up to each one to define what part of the system that needs more attention than the rest. There is though an agreement to keep the TCB as small as possible, as it is with the kernel. The smaller; the easier to manage. Not all of TCB has a kernel. Some part or the whole OS can be part of TCB. When the TCB is defined in a system it is then easier to relate to the "TCB" in the security policy, like they do in the "Department of Defence trusted computer system evaluation criteria" [59]. For different areas like; "Discretionary Access Control", "Identification and Authentication" and "Operational Assurance" they write: "The TCB shall". So they only make a reference to the TCB and the TCB is defined some where else.

Peer to peer systems and BitTorrent do not offer its users anonymity except for Thor and The FreeHaven project which take a lot of bandwidth. It is possible to obtain the IP addresses of the connected computers or swarm, having the piece of the file that you want in BitTorrent clients. The tracker is the piece locating the swarm and it has then the information about the IP addresses in the swarm of computer nodes. BitTorrent uses cryptographic hashing of all data which makes it harder to detect BitTorrent files that Internet Service Providers (ISP) like to filter out. The data then only look like a hash function and will not give away any information that the file is really a .torrent file.

Trust in peer to peer networks does work like trust among people in standard networks. In the client applications a peer can get trust by how much the peer has contributed to the system or also being recommended by others. Online shopping sites with links to many online stores publish the views of a person using a particular online store. The feedback could be if the service was bad like it took long before the item was sent to them or if the store is super good. Usually the online store gets an amount of stars behind its name on the online shopping site.

4.3 Physical security

Using a distributed system will make the user less vulnerable to server crashes or floods from water or earth quakes. In a distributed system it does not matter if a node is down, since the files are spread on many nodes. The nodes that are left are enough for the file to be rebuilt [51]. The network naturally tolerates hardware faults when nodes are organized in a non-hierarchical way like with peer to peer systems and each node only holds a list of its neighbours instead of a long list of many peers. A long list of peers will only slow down the tracking of the nodes. Unfortunately the peer to peer network can not guarantee communication between nodes, by using the before mentioned Information Dispersal Algorithm (IDA) in 2.1.3.2 we can get a more secure and fault tolerant transmission and with a better load balancing [52].

In a distributed system encryption is used to hide the file extension for Internet Service Providers (ISP) so that they will not filter out the data. It is also used for anonymity so that peers in totalitarian countries like China and Iran can communicate with other peers without being sent to prison. For our system, we will use encryption to preserve privacy or integrity of the data.

Distributed Internet Backup System (DIBS) [29] is based on cheap storage space and so should backups be. The data is encrypted using Gnu Privacy Guard (GPG) which also supports smart cards. DIBS use the error correction code called Reed-Solomon which means here that each file you want to backup is split into N pieces. To recover that file, you only need N/2 pieces. With a peer to peer network of 14 nodes, 7 nodes could be disconnected and you can still recover the file. This sounds good for physical security for your stored files. The RAID backup system can only rebuild a file if one node/disk is down.

5 DESIGN OF HYBRID ARCHITECTURE

So what is the perfect design of a backup distributed system when it comes to resource location and security? What topology will be the best to use and what search protocol.

5.1 Topology

If NASA thinks that JXTA is good enough for them, I think it is good enough for my hybrid architecture also. Both Nokia and the Jet Propulsion Laboratory (JPL) seem to be able to adopt the JXTA to their need. Nokia writes that it is easy to add an additional server using the JXTA advertisement and discovery service. They also perform their database replication and synchronization using the JXTA socket which is an Application Programming Interface (API). JPL is more into the query part of the JXTA protocol. So, I can use JXTA for addressing, routing and messaging. Also I can do discovering, grouping and communication between pc's and devices [39]. The only thing I need to do is to program JXTA as I want it to be.

There was a concern about using a peer to peer topology since peers are believed to be lying about their characteristics like bandwidth and computing power. I do not see this as a problem since there are ways of providing this information using mechanisms in the network. Backup network storage is a bigger problem than bandwidth and computing power [70]. A peer can though be required to publish auditable records and also allowing other nodes to audit the peer. Also agents can flow in the network checking disk space.

5.2 Protocols and algorithms

No extra hardware is needed for the independent users on the internet to join up and share files. We use the internet and on top of that we put a protocol forming an overlay network as with the persons in the figure below.

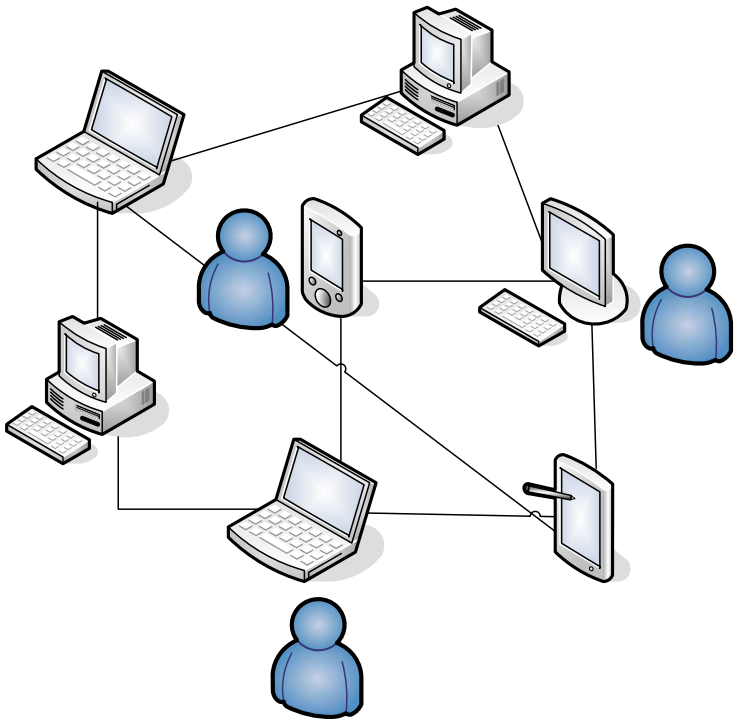

Figure 12 Internet with a peer to peer overlay network

Napster and Gnutella form an overlay network within the internet, but the protocols used in Napster and Gnutella are not good enough. Napster uses the HTTP internet protocol with a central server and Gnutella uses flooding in its protocol. There are other protocols available which are better. The protocol should also support an ad-hoc network since so many are using small devices like mobile phones today.

A Content Addressable Network (CAN) is a Distributed Hash Table (DHT), but will not be used in this network since it is best used with data that do not change often. As a user I will probably update my files quite often. A User Datagram Protocol (UDP) using a XOR based metric topology with the Kademlia protocol which also is a DHT algorithm, seems to be a good choice. Node look-ups can be done asynchronously with Kademlia. Kademlia is used for file sharing and used with BitTorrent.

The de Bruijn topology using the Broose protocol is also a peer to peer network using the DHT algorithm, but maintained in a loose manner, better than Kademlia. The protocol allows load balancing of hot spots which are public accessible wireless connections to the internet [50]. This is the first protocol handling key collisions. Its goal is to be as practical as Kademlia. JXTA is another overlay network and uses different protocols for the different set of peers in the network. There are edge-peers and Super-peers and they have their own rules or protocols to follow. It is the same with the peer behind the firewall. DHT is applied when the rendezvous or Super-peer is forwarding an advertisement index to another peer.

The only protocol supporting ad-hoc devices and is fast on the query side, is the JXTA protocol. I will not use any other protocol since JXTA can be used with both TCP and UDP.

When storing a file in the peer to peer network I would like it to be split so that it can be partly stored on different peers. There are different ways of doing this. The Information

Dispersal Algorithm (IDA) is a protocol which breaks a file into pieces, adds an overhead of data so that some of the pieces can be used to reconstruct the data afterwards. In [53] they propose to use IDA for signature authorizing instead of a Tornado type code, because Tornado is said to be better for large segments to code. IDA is though better when the number of segments is smaller. We assume that the number of segments that we have is going to be small compared to using Tornado, RS code or LT code.

JXTA has all this already and the possibility for encrypting files and applying a key. JXTA is supposed to handle any type of files, but I do not know if it sends a whole Codat (data) or part of one. I will use JXTA here also since it looks like that JXTA can give me "the whole packet".

5.3 Security

In a peer to peer network, your access to the network resources will not be jeopardized by some few nodes failing or not being connected. Centralized systems are vulnerable to local catastrophes. Large companies have backup storage places spread around the world to avoid this. By using peer to peer networks, this problem can be avoided.

How secure should the data be? The personnel of Microsoft suggest a serverless file storing scheme without mutual trust [68]. They say that without mutual trust we need file crypto and digital signing of the file when the file is updated. They propose that the file should be encrypted before they are replicated. They also worry about encryption interfering with duplication control. They propose though to do a one-way cryptographic hash and sign the hash which corresponds to what other systems also do.

When the file is finally sent away to some peers, then I would like to use reverse sandboxing [48]. If I want to store a file on an area you have opened up for me on your computer which is what you do with file sharing software, I want my files protected from being tampered with from you and your machine. My file will be pre-encrypted using a cryptographic file system like the one that was proposed in "Efficient and Secure Information Sharing in Distributed, collaborative Environment". When the file needs updating it will not be updated. The whole file will be changed out and also the key belonging to it. My client software and the client software running in the background on your computer will exchange keys using public key infrastructure (PKI) or Gnu Privacy Guard (GPG). There will be no authentication or control on the shared hard disk because *transitive delegation* [4] [48] is used. Transitive delegation is showed below. I have created a delegate on a shared area on Peer 1's computer. Peer 1 has also created a delegate on my computer. We though do not necessary trust or know of each other.

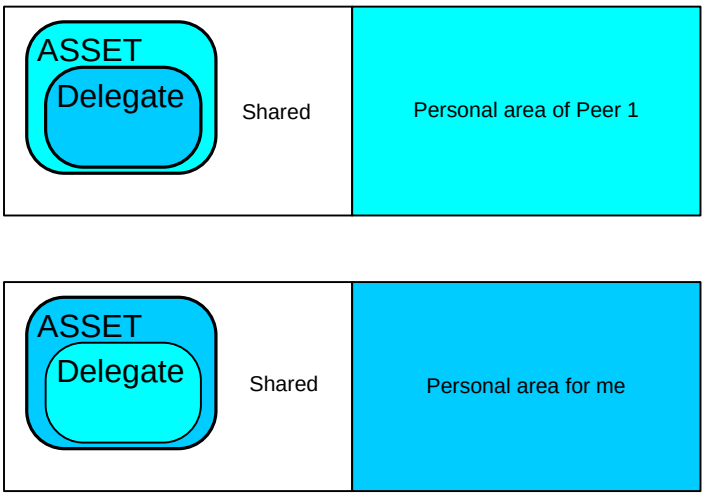

Figure 13 Transitive Delegation

Confidentiality will normally be provided by the use of keys. Symmetric keys will be used for cryptography where the symmetric key is shared beforehand using public key cryptography. AES can be used for the symmetric part and RSA for the public key cryptography [53] part. It is possible to have a single signature scheme over multiple packets, so this will be used.

6 DISCUSSION

6.1 The 3 backup systems

"A Survey should be made of different current architectures for resource discovery in large scale systems, such as in popular p2p systems"

6.1.1 Efficient and Secure information sharing

Even though the paper on "Efficient and Secure information sharing" came out in 1999, it still seems very new. It has some surprising ideas that none of the other two bring in to their paper. One of them is the idea of using "Sandbox". The paper focuses on "Efficient and Secure information sharing", but does not mention any look-up service. I assume the machines sharing the information already know of each others IP addresses etc, so standard routing is enough. In the paper they mention secure storing and key handling which is a good thing and the collaboration mode is similar to CORBA, so I do not understand why CORBA can not be used directly. The key handling seems a bit cumbersome. They rely on a "key server". I assume that when they mention "file server" and "key service" this is hopefully not something that needs to be centralized. A "file server" and a "key service" can be objects distributed to different machines. I though like the idea of changing out the whole file and the key, when a file needs an update. Personally it scares me to have different versions of a file. Also Microsoft mentioned that the duplication of files can be a problem. As for explaining the file size, they only suggest to divide the files into short physical files instead of storing the whole large file. They also suggest distributing these short physical file on different file servers. I assume a file server can be a standard machine. These views fit with what I find is important in an architecture.

The crypto file system they suggest will separate authentication and access. Here there is a separate system encrypting the files and a different system actually signing the file instead of one system encrypt and then sign in one go. I do not like the use of CA generally, because it smells like money. The CA could though be an internal CA for the peer to peer network.

6.1.2 pStore

pStore uses Chord to do the look-ups. The only reason for this must be that Kademlia [65] was released the year after, which was in 2002. Kademlia has a better distance metric for the nodes. Nodes can not be added in Chord [66] before Chord has finished its round updating which nodes are still in the network. With Kademlia a node can be updated if a node is querying another node. There is no separate update method in Kademlia as it is with Chord. I though think that the versioning system CVS which is vital for the open software groups, is a good idea. CVS seems to work for multiple code writers spread across the world, updating the same application. The pStore group was running tests with and without versioning, both using Unix rsync and CVS for versioning. Tests showed that CVS used more bandwidth than the others at first, but the cause of this was that it was doing a full backup of all files. Later when only doing updates of the files, it used less bandwidth. Whatever versioning system

chosen, at least I think it is a good idea using something that is already working. Both rsync and CVS have proved that they are working. Block sharing used in pStore does also help keeping the bandwidth lower both when using versioning or not. The blocks of data can be viewed as an idea of splitting the files in to smaller pieces which has already been suggested done earlier.

6.1.3 iDIBS

iDIBS use LT erasure coding to restore data in their Distributed Internet Backup System. According to [53] "Efficient Multicast Stream Authentication Using Erasure Codes" Tornado coding, which came before the LT coding can not be used for their purpose. Tornado and LT are effective on large number of segments which will work for iDIBS. RS, Tornado and LT coding can be used here since the purpose is to reconstruct large number of files and not signing small packets of data. Instead of using IDA to be able to sign multiple packets, iDIBS use Gnu Privacy Guard (GPG) for both encryption and signing. iDIBS do incrementing backups meaning that it updates a file only if there has been a change to it. Just part of the file needs updating. The files are split into blocks which make the overall performance better as done in the other schemes.

6.2 Security requirements of a system

"There is a security aspect in using a Distributed System for storing your personal files; confidentiality, integrity and availability need to be taken into account"

6.2.1 Topology

The topology of the system is important for the *availability* of the resources in the system. A peer to peer topology is more secure than a centralized topology because of geographically localized faults as for natural disasters with failed routers or broken network cables [68]. The uptime of a peer to peer system is very high because of the redundancy. By using a hybrid or peer to peer network topology we will also save money on expensive servers and network administrators. A peer to peer system is always available for any users if there are no security algorithms in the system.

The JXTA seem to be a very good system to choose since security is built into it from the start. Since my architecture needs a "key server" the Super-peer will have enough capacity to handle this job. Unfortunately the certificates the system is using will have to be backed up outside the peer to peer network.

6.2.2 Algorithms and protocols

As for data encryption and signing the data, the old basic strategies are still valid, securing the *confidentiality* and *availability*. Encrypting the data while still on disk is not a problem even though cryptography steals a lot of CPU power. Signing data also takes some computation,

but a least if the signing is done just once per file it is less costly. Trust was an issue, but both Microsoft and the "Efficient and Secure information sharing" do not see the need for this. In a peer to peer network as in interpersonal networks, a kind of trust is built up over some time. Encryption and signing of data should be enough, but in addition to this the architecture will use the "reverse sandbox" technique. This will make the data on another peers' machine even safer also using transitive delegation will able another person that I trust to access my data.

6.3 Design of new system

"Investigate/design a hybrid architecture which achieve reliable resource discovery with minimal centralized server requirements, i.e. which aims to reconcile the favourable properties of both distributed and centralized systems"

6.3.1 Topology

Unfortunately I did not understand the quality of JXTA until I had surveyed other peer to peer architectures first. I was a bit unsure for a while if JXTA should have gotten a more prominent place in my thesis and after choosing JXTA I see that it should. The layers in the JXTA protocol with Super-peer and standard nodes on one level and an application user interface on the next level is really impressive. Anybody can make JXTA fit their purpose and policies which is just what a perfect tool is about.

6.3.2 Algorithms and protocols

The possibilities for the JXTA and its protocols seem to be unlimited. JXTA has a number of protocols for the individual status of the peers, like the Super-peers will use two different protocols which is one for the rendezvous peer and one for the relay peer allowing companies accessing distributed machines behind a firewall. DHT is a very important algorithm as well and JXTA uses it.

7 CONCLUSIONS AND FURTHER WORK

7.1 Conclusions

I looked into the security aspects related to using distributed systems for resource discovery and I have suggested a design of a resource discovery architecture which will use distributed systems for backup of personal data with minimal centralized server requirements.

I began looking into the theory and the state of the art technology related to anything on peer to peer networks and resource discovery. Resource discovery in peer to peer systems can be used not only for file sharing and backup, but the game industry is also very interested in it. I did not have to look into real-time data, which can be challenging. Static resources like files are more manageable.

Today's protocols have already shown their qualities which made it possible to investigate an imaginary perfect backup system. The only way of knowing if JXTA can be used for this is to actually write Java code with it.

I have answered the questions for the requirements of a perfect backup architecture.

The questions from chapter 1.3 with answers:

- The system will find my file and not some other persons file

Using JXTA and the protocols associated with it, I will find my file and not some other persons' file.

- Time is not so important

Finding my file will be done in a timely matter. JXTA "search" suits environments where content is rapidly changing [38].

- No one else should be able to read my file

If I am using a cryptographic file system like it was suggested in the "Efficient and Secure Information Sharing in Distributed collaborative Environment", with digital signing of the file with a key, I will keep anyone else from reading my file.

- I can share a file with someone I trust

By using transitive delegation I can as an agent, create a delegate which can access certain objects, which here will be my file.

- No one else should be able to change my file
- A person I trust can change the file

A person that I trust can then access the file, either as read-only or execute. Others will not have access. Since I am using the JXTA configuration, the key associated with the file will be handled by a key server which is one of the Super-peers. For extra security the area on the other peers' computer is reverse sandboxed. The other peer is not trusted.

- If my unit goes down I will be able to retrieve all backed up files

The key for retrieving files is stored on a key server. The certificate I will need to backup to some where safe, so that I can use the backed up certificate to retrieve my files if my computer dies.

- It must be free
- Enough storage space

This set-up is using open source software and is free and according to Microsoft and the Google founders [68], [62] there is an overhead of storage space on the internet.

7.2 Future work

Since I did not have time to test the different systems, it would be interesting to see what they can do, especially the JXTA protocol. It took me a while to find that it is a very popular protocol. There has been done a lot of work on it and the only way of getting to know is to play with it. Looking at the JXTA home pages, it looks like a very versatile protocol.

ABBREVIATIONS

ACL	Access control list
Ad-Hoc	Unplanned (here for a network)
AES	Advanced Encryption Standard Up to 256 bits
CA	Cartificate Authority
CAN	Content Adressable Network
CBAC	Context based access control
CSOAP	Client SOAP
CORBA	Common Object Request Broker Architecture
CVS	Concurrent Versions system, open source ver control
DAC	Discretionary Access Control
DES	Data Encryption Standard 56 bits
DHT	Distributed Hash Table
DIBS	Distributed Internet Backup System
DRM	Digital Right Management
GPG	Gnu Privacy Guard
Hotspot	public accessible wireless connections to the internet
IDA	Information Dispersal Algorithm
ISO	International Organization for Standardization
ISP	Internet Service Provider
LSI	Latent Semantic Indexing
MAC	Mandatory Access Control
OS	Operating System
OSI	Open System Interconnection
PDA	Personal Digital Assistance
Peer	a computer node with the same status as you
PKI	Public Key Infrastructure
P2P	Peer to Peer
QoS	Quality of Service
RSA	Rivest, Shamir and Adleman. Public key crypto
RBAC	Role Bases Access Control
RS code	error correcting code
Reversed Sandbox	Protecting my programs from your machine
Sandbox	Protecting my machine from your programs
SOAP	Simple Object Access Protocol
SLP	Service Location Protocol
SWAN	Small World Adaptive Networks
SWAN 2	Stateless wireless Ad Hoc Networks
Swarm (P2P network)	connected computers having part of a file

TCB	Trusting Computing Base
TCP	Transmission Control Protocol
Two-tier	Two-layer
UDDI	Universal Discription Discovery and Integration
URL	Uniform Resource Identifier
VPN	Vertual Private Network
XOR	negate of OR returning '0' when bits are the same

REFERENCES

[1] Context-Aware Computing. Thomas P. Moran and Paul Dourish. 2001.
[2] Distributed systems, Principles and Paradigms. Andrew S. Tanenbaum and Maarten van Steen
[3] http://www.laas.fr/mosaic/documents/storage-edcc-2006/storage.html
[4] PEET-TO-PEER COMMUNITIES: ARCHITECTURE, INFORMATION AND TRUST MANAGEMENT Mujtaba Khambatti, 2003.
[5] http://delivery.acm.org/10.1145/1150000/1141672/p1669-jin.pdf?key1=1141672&key2=5916908711&coll=ACM&dl=ACM&CFID=17793268&CFTOKEN=86871692
[6] Content-Aware search of Multi Media data in Ad Hoc networks, Bo Yang and Ali R. Hurson, 2005
[7] A Taxomony of Discovery Services and Gap Analysis for Ultra-Large Scale Systems, K Krauter, R Buyya, M Maheswaran
[8] http://people.cs.uchicago.edu/~anda/papers/thesis-abstract.html
[9] http://bfi-internal.org/sustainability/node/68
[10] http://www.sequoiabroadband.com/technology_centralized.html
[11] http://acronyms.thefreedictionary.com/
[12] http://www.gris.det.uvigo.es/~rebeca/vodca/slides/Bella_Pistagna_Riccobene_vodca04.pdf
[13] http://www.inria.fr/rapportsactivite/RA2005/arles/uid34.html
[14] http://www.openp2p.com/pub/a/p2p/2001/12/14/topologies_one.html
[15] A Semantic Search Framework in Peer-to-Peer based Digital Libraries. Hao Ding. 2006
[16] http://www.vpnc.org/ietf-ipsec/92.ipsec/msg02350.html
[17] http://www.microsoft.com/industry/financialservices/insurance/businessvalue/wifpim.mspx
[18] http://www.inria.fr/rapportsactivite/RA2005/arles/uid19.html
[19] http://autoidlabs.mit.edu/whitepapers/MIT-AUTOID-TR-004.PDF
[20] http://citeseer.ist.psu.edu/jun99agentbased.html
[21] Resource Discovery in Distributed systems. Mor Harchol-Balter, Tom Leighton and Daniel Lewin. 1999.
[22] http://wpage.unina.it/cotroneo/dwnd/P2P/P2P_DHT.pdf
[23] Efficient query routing for Information Retrieval in Semantic Overlays. Hai Jin, Xiaomin Ning, Hanhua Chen, Zuoning Yin. 2006
[24] http://privacy.med.miami.edu/glossary/xd_confidentiality_integrity_availability.htm
[25] The MSDA Multi Protocol approach to Service discovery and access in pervasive Environments Pierre-Guillaume Raverdy, Rafik Chibout, Agnès de La Chapelle and Valèrie Issarny 2005
[26] A Security Focused Integration Architecture for an Electronic Observation ChartInformation Security. NTNU. Divic, Mirela, Huse, Hveding. 2005.
[27] Distrubuted Role Based Access Control for Dynamic Coalition Environments Eric Freudenthal, Tracy Pesin, Lawrence Port, Edward Keenan and Vijay Karamcheti 2002
[28] http://www.ssh.com/support/cryptography/protocols/
[29] http://web.mit.edu/~emin/www/source_code/dibs/index.html
[30] http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.express.doc/info/exp/ae/cwbs_wssmessage.html

[31] http://emhain.wit.ie/~pwall/CvD.htm
[32] http://www.docuverse.com/blog/donpark/2003/11/30/how-bittorrent-works
[33] http://delivery.acm.org/10.1145/1240000/1233406/p355-hoffert.pdf?key1=1233406&key2=7560108711&coll=ACM&dl=ACM&CFID=21341166&CFTOKEN=60501093
[34] P2P systems based on distributed hash algorithm. Ming Xie. 2003.
[35] http://www.inria.fr/rapportsactivite/RA2005/arles/uid15.html
[36] http://linuxjournal.com/article/6797
[37] http://www.bittorrent.org/Draft_DHT_protocol.html
[38] http://www.jxta.org
[39] http://java.sun.com/developer/technicalArticles/Networking/jxta2.0/
[40] www.ariadne.ac.uk/issue8/resource-discovery
[41] http://technet.microsoft.com/en-us/library/9c9e0367-4032-47be-b334-bf3fed9ea539.aspx
[42] http://www.cs.washington.edu/homes/gribble/papers/mmcn.pdf
[43] Centralized versus Decentralized Computing: Organizational Considerations and Management Options. John Leslie King. 1983.
[44] http://ieeexplore.ieee.org/iel1/49/11144/00508287.pdf?tp=&isnumber=11144&arnumber=508287
[45] http://www.cs.vu.nl/~bpopescu/papers/cms02/node1.html
[46] http://delivery.acm.org/10.1145/170000/165613/p6-mirhakkak.pdf?key1=165613&key2=7301515711&coll=ACM&dl=ACM&CFID=18557190&CFTOKEN=77586152
[47] Computer security technology planned study. Jampes P. Anderson. 1972.
[48] Efficient and secure information sharing in distributed, collaborative environments . Partha Dasgupta, Vijay Karamcheti and Zvi M. Kadem 1999.
[49] D2B: a de Bruijn based content addressable network Pierre Fraigniaud and Philipe Gauron
[50] Broose: A practical distributed hashtable based on de Bruijn topology 2004. Anh Tuan Gai and Laurent Viennot.
[51] Distributed Backup through information Dispersal. Giampaolo Bella, Costantino Pistagna and Salvatore Riccobene. 2004.
[52] Information Dispersal and Security, load balancing and fault tolerance. Michael O. Rabin. 1989.
[53] Efficient Multicast Stream Authentication Using Erasure Codes. Jung Min Park, Edwin K. P. Chong and Howard Jay Siegel. 2002.
[54] PStore: A Secure Peer to Peer backup system. Christopher Batten, Kenneth Barr, Arvind Saraf and Stanley Trepetin. 2001.
[55] http://sourceforge.net/projects/dibs
[56] iDIBS: An Improved Distributed Backup System. Faruck Morcos, Thidapat Chantem, Philip Little, Tiago Gaiba and Doug Thain. 2006
[57] dRBAC: Distributed Role-based Access Control for Dynamic Coalition Environments, Eric Freudenthal, Tracy Pesin, Lawrence Port, Edward Keenan and Vijay Karamcheti. 2001.
[58] The Role of Trust Management in Distributed Systems Security. Matt Blaze, Joan Feigenbaum, Hohn Ioannidis and Angelos D. Keromytis. 1999.
[59] Department of Defense trusted computer system evaluation criteria. 1985.
[60] Security in distributed peer-to-peer systems. IKT 404. Borgi, Isfeldt, Larsen. 2006.
[61] Quality of Service (QoS) in mobile ad hoc networks. Morten Kronstad Vinje. 2006.

[62] http://infolab.stanford.edu/~backrub/google.html
[63] Access Control in Distributed Object Systems: Problems with Access Control Lists. S. V. Nagaraj. 2004.
[64] Rarest first and Choke algorithms are enough. Arnaud Legout, G. Urvoy-Keller and P. Michiardi. 2006.
[65] Kademlia: A Peer-to-peer Information System Based on the XOR Metric. Petar Maymountkov and David Mazières.
[66] Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek and Hari Balakrishnan. 2001.
[67] http://delivery.acm.org/10.1145/950000/941154/6797.html?key1=941154&key2=0349510811&coll=ACM&dl=ACM&CFID=19659166&CFTOKEN=64261270 DHT part I.
[68] Feasibility of a Serverless Distributed File System Deployed on an Existing set of Desktop PCs. William J. Bolosky, John R. Douceur, David Ely and Marvin Theimer. MS research. 2000.
[69] Guarding Security Sensistive Content using Confined Mobile Agents. Guido van `t Noordende, Frances M.T: Brazier and Andrew S. Tanenbaum. 2006.
[70] Economic Behaviour in Peer-to-Peer Storage Networks. Andrew C. Fuqua, Tsuen-Wan Ngan and Dan S. Wallach. 2003.
[71] http://www.pcworld.dk on JXTA. 2001.
[72] gnunet.org
[73] howstuffworks.com
[74] wikipedia.org
[75] gnutella.com